Improving architectural 3D reconstruction by constrained modelling

Helmut Cantzler

Doctor of Philosophy

Institute of Perception, Action and Behaviour
School of Informatics
University of Edinburgh

2003

Abstract

This doctoral thesis presents new techniques for improving the structural quality of automatically-acquired architectural 3D models. Common architectural properties such as parallelism and orthogonality of walls and linear structures are exploited. The locations of features such as planes and 3D lines are extracted from the model by using a probabilistic technique (RANSAC). The relationships between the planes and lines are inferred automatically using a knowledge-based architectural model. A numerical algorithm is then used to optimise the position and orientations of the features taking constraints into account. Small irregularities in the model are removed by projecting the irregularities onto the features. Planes and lines in the resulting model are therefore aligned properly to each other, and so the appearance of the resulting model is improved. Our approach is demonstrated using noisy data from both synthetic and real scenes.

Acknowledgements

Firstly I would like to thank both my supervisors Robert B. Fisher and Eric McKenzie for guiding me through my PhD and the members of the Computer Vision group at Edinburgh university, in particular Neil McCormick for his valuable advice during writing up. Furthermore, I thank all members of the European research network CAMERA (ERB FMRX-CT97-0127), with whom I worked together closely. CAMERA undertook crossdisciplinary research in the field of automated acquisition of 3D models of existing architecture. I especially need to thank Michel Devy for his support during my three months stay at the Laboratoire de Analyse et de Architecture des Systemes (LAAS-CNRS) in Toulouse. Many thanks to my friends who supported me during my work, in particular to my good friend Robert Redford with his deep knowledge about architecture.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Helmut Cantzler*)

Table of Contents

Chapter 1

Introduction

Recovering the 3D structure of the environment we are in is a constant experience in everybody's life. We recover the depth of a room we are standing in or the outdoor structure of a building we are looking at. We do this by extracting the depth of the scene with our eyes. They function as a stereo system to extract the depth of the scene which is in our view. This system works as a continuous process. But even without using our 3D perception (for example if we close one eye) we are able to recover the 3D structure. We do this by analysing motion in the scene or applying higher knowledge to our image of the scene. Changes in texture show us borders of different surfaces while straight lines show edges in the 3D structure such as the transition from a wall to the ceiling. After briefly looking around we are able to model the 3D structure of a building in our mind.

Computer vision tries to imitate this human ability of perception in order to reconstruct 3D models. Indeed, 3D reconstruction is one of the hardest major applications of computer vision. Many different approaches exist to perform the task. Many researchers have concentrated work on reconstruction of buildings [DTM96, FRL+98, Pol99, DTC00, WZ02a], because of their common appearance in our everyday life.

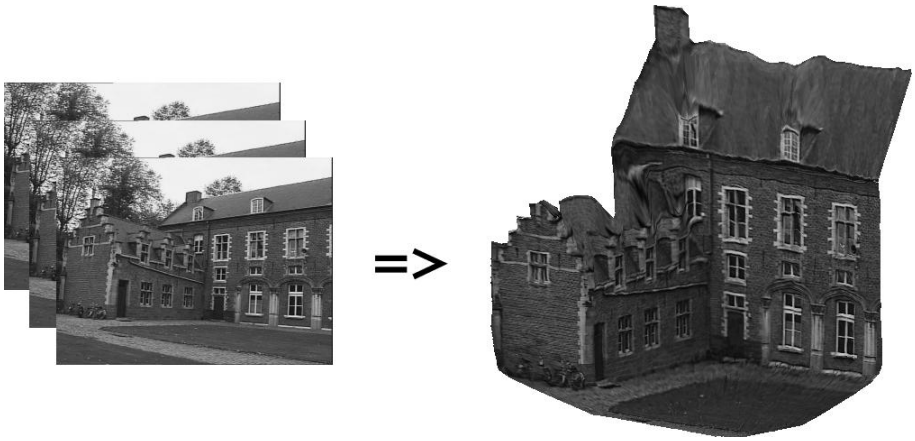

Figure 1.1: An image sequence of 20 images (left) of Arenberg castle (Belgium) was used to reconstruct a textured 3D model (right). The reconstruction was carried out by the Catholic University of Leuven.

1.1 Reconstruction of built environments

Reconstructing the 3D structure of an existing building is useful in many different areas. A reconstruction may be used to create CAD models, where CAD models are not available or out of date. 3D models of sensitive buildings (like nuclear plants) are used for monitoring and modification planning. Training facilities need accurate 3D models for virtual reality training scenarios as part for their training repertoire. Tourist, museum and archaeological sectors use 3D models of valuable historical sites for promotion, public education and preservation. An important area for environment reconstruction is robotics. The 'Holy Grail' for mobile robotics research is to design a robot that can function in a real world environment [WSV99]. As a prerequisite, the robot must know the structure of the environment it is in to navigate properly.

Reconstruction techniques based on photogrammetry use a set of images from a scene. Small strong reflective points (so-called 'targets') are placed in it. With the help of these, correspondence between the different scene images can be found easily. Stereopsis can then be used to calculate the 3D structure of the scene and triangulation is applied to obtain a representation of the scene as a triangular mesh. In general photogrammetry techniques mostly focus on problems where high accuracy is needed

[Atk96]. However, they generally require heavy human interaction.

In computer vision, techniques that work more automatically are used, which do not require targets or known camera parameters [DA89, JJ91, Fau93, FRL+98]. The reconstruction can be done with a whole array of different techniques known generically as Shape-From-X. Much work has been done using intensity images for the reconstruction, because the equipment used to capture the scene is cheap. Often multiple images taken from different viewpoints are used. The equivalent to photogrammetric targets are the 'feature points' in the images, which are harder to extract from a scene. The 3D structure of the scene is calculated using stereoscopy without necessarily requiring the camera parameters. However, in general, automatic detection of features and their correspondence in a sequence of images can be an error prone process. The resulting 3D model is less accurate than they would be with photogrammetry techniques and may contain gross errors. An example of a reconstructed building is displayed in figure 1.1. The textured 3D model was created by the Catholic University of Leuven [Pol99].

1.2 Motivation and aim

The process of architectural 3D reconstruction of scenes is often affected by noise in the sensor data. Furthermore, inaccuracies are created by view merging, distortions in the data and surface fitting. Figure 1.2 shows a range data scan of the front of the central Edinburgh mosque (left) and a model of a Bavarian farmhouse (right) reconstructed by the European Commission Joint Research Centre [SNW+99]. The range data scan shows the entry of the mosque on the right, the base of the minaret in the middle and parts of walls on the left side. One can clearly see that this scan is very noisy. The image of the Bavarian farmhouse is a close-up view of the farmhouse wall. One sees surface irregularities in the form of surface ripples between the windows.

The principal objective of this study is to overcome the above inaccuracies. Examples of ways to improve the reconstruction are to use more sophisticated methods such as photogrammetry techniques, to increase the number of views or to combine sensor data from different sensors like range scanners and digital photo cameras

Figure 1.2: The left image shows a raw range data scan. At the top is a doorway and several walls. The scan also includes a serious amount of noise (cloud of grey dots). The right image shows a close-up of one wall of a farmhouse reconstructed by the European Commission Joint Research Centre. Surface ripples between the windows are easily seen in the circled areas.

[WWM01, BGCA02]. However, these solutions are work intensive, expensive or difficult to carry out.

We propose instead a methodology aiming to improve the quality of reconstructed models by identifying and exploiting architectural scene properties. 3D reconstruction of industrial parts [RFWA00, WFAR99b, MLMM01, LMM02] and image based 3D reconstruction of architecture [FRL+98, CRZ00, DTRC01, WZ02b] have involved use of prior knowledge to improve models for some time. Chapter 2 presents a full length literature review with a detailed analysis of the use of prior knowledge in the form of constraints in 3D reconstruction and modelling. Architectural scenes are particularly suitable for the application of constraints since the geometry is typically very structured [Chi96] (see Chapter 3). This thesis therefore proposes a process to improve the structural quality of automatically-acquired architectural 3D models by applying constraints.

We developed a three stage process. The first stage consists of extracting archi-

tectural features such as planes (*e.g.* walls) and lines (*e.g.* linear structure such as window outlines) from the model data (see Chapter 4). The parameters of these features build up the description of the 3D model. The second stage is the interpretation of the scene (see Chapter 5). We automatically discover the relationships between the features and therefore the constraints to enforce orthogonality and parallelism between the planes and lines. The last stage is the optimisation of the model description so that all constraints are satisfied (see Chapter 6). For example, two walls are properly aligned parallel to each other. The technique also aims to remove small irregularities on planes and jitter on lines. The resulting model with corrected feature orientations and removed irregularities is created by projecting the model points onto their optimised positions. A primary requirement is that both the constraints in the optimised model are satisfied and the optimised model resembles the original model closely. We evaluate our methodology with models created from both synthetic and real data (see Chapter 7).

The research contributes to the field of environment reconstruction by exploring the use of architectural knowledge of built environments including both interiors and exteriors. It postulates that constraints which use this knowledge will resolve the effects caused by noise, and so the accuracy of the reconstructed 3D architectural model will be improved.

Chapter 2

Literature review

3D models are in fact becoming increasingly visible in our lives these days. They appear frequently in cinema, TV, computer games, art and medicine. One reason is the recent massive increase in computer power and in particular the availability of specialised yet cheap 3D display hardware. Another reason is the recent improvements arising from research done in 3D modelling.

As we mentioned in Section 1.1, several research disciplines including photogrammetry, computer graphics and computer vision deal with the creation of realistic 3D models. The models are reconstructed from all kinds of different sensor data such as range data (taken from laser, sonar or radar sensors), sets of photographs or even a single photograph. Techniques based on photogrammetry mostly focus on problems where high accuracy is needed [Atk96]. However, they generally require heavy human interaction. An operator marks the scene with small strong reflective targets. With the help of the targets, accurate correspondence between the photographs of the scene taken from different viewpoints can easily be found. In computer vision, researchers use different, more automatic, techniques for acquisition of 3D models [FRL+98, DA89, Fau93, JJ91]. The computer vision equivalent to photogrammetric targets are feature points, which can be for example strong corners or line intersections in photographs. Finding the same feature in different photographs of the same scene means a correspondence is established. Detection of features and an accurate correspondence between features are both harder to obtain with computer vision than with photogrammetry. Therefore, reconstruction of 3D models using computer vision tech-

niques is not as robust, the result is less accurate and may contain gross errors. However, computer vision solutions have the advantage of being much less work-intensive (*e.g.* no markers need to be placed in the scene).

Many techniques have been proposed to increase the accuracy of the reconstruction. The first thing one could do is to increase the resolution of the data, using a more precise sensor like a range finder, or by combining data from different sensor types. However, these solutions are often more difficult (*e.g.* fusing data from different sensors) and expensive (*e.g.* more precise sensors are typically more expensive). In this chapter we deal instead with how one can exploit prior knowledge about the scene to improve the 3D reconstruction.

Many man-made scenes include symmetric structures. 3D model reconstruction from an image sequence can be improved by exploiting bilateral symmetry [ZW97]. Symmetric objects are extracted from the images by searching for the closest configuration (group of points) in a least-squares sense. For this purpose, the symmetry distance is defined for any configuration of points with respect to any other configuration of points. The symmetric groups found are then constrained in the reconstructed 3D model.

Many man-made scenes are also very structured in ways other than symmetry. Scenes containing industrial parts or buildings are good examples. Often prior knowledge of the scenes exists (*e.g.* two planes are parallel or they meet at 45 degrees). This prior knowledge can be applied to the model to constrain it. One way to do this is to use parameterised object models and to fit the original data to the object model. The other more general way is to constrain a certain part of the scene. A constraint can be used to enforce the shape of a model part or to enforce a certain relationship between two parts of the model. The appropriate application of constraints leads to a more accurate model [WFAR99b].

We concentrate here on the use of constraints to improve the structural quality of the 3D reconstruction. Improving the quality means both improving the precision and the appearance. We want to constrain typical architectural features such as planar surfaces and linear elements (*i.e.* lines). The process of application should be automatic as much as possible.

In this chapter, we will first review which 3D representations have been used to represent architectural structures. We then discuss research which has used geometrical constraints to model both generic objects in section 2.2 and in particular architecture in section 2.3. The main focus here is on 3D reconstruction of buildings.

2.1 3D representations

The ways in which we can represent 3D objects in an environment are almost as many and varied as the objects themselves. We can choose from a wide range of techniques and data structures, which differ in run time and accuracy, to represent 3D objects.

The choice of a representation depends on the scene we want to model. The natural environment is more difficult to represent, because it is typically rather unstructured. As a consequence, the use of simple low level representations like polygon meshes is more suitable than the use of representations based on geometric primitives [DCF+94]. On the other hand, more structured scenes like architectural scenes can be nicely represented by a combination of geometric primitives. Computer aided design (CAD) systems often use this representation. 3D representations for mobile robots can use more symbolic information like a topological representation which separates terrain areas into regions [DP99].

The following sections describe the representations commonly used in architectural modelling. The representations are presented in order (loosely) of frequency of use. Bi-cubic parametric patches and spatial subdivision techniques are not included since they are not frequently used to model architecture.

2.1.1 Polygon mesh

The most popular representation for architectural modelling is undoubtedly the triangular mesh as a form of a polygon mesh [SGR95, Toe96, FRL+98, Pol99, RC00, DTC00]. The polygon mesh is the classical representation in 3D computer graphics [Wat00]. Objects are represented by a net or mesh of planar polygonal facets. The facets are an approximation to the curved surfaces of an object. With this form we can model, to any accuracy that we choose, an object of any shape. We can constrain all

polygons to be triangles to gain improved performance for rendering with 3D graphics accelerator cards.

Polygon meshes are a machine representation - rather than a convenient user representation - and they are often used for other representations which are not directly renderable. Thus bi-cubic parametric patches, constructive solid geometry and voxel representations are often converted into polygon meshes prior to rendering.

One of the significant developments in 3D graphics was the emergence in the 1970s of algorithms, like shading algorithms [Pho75], that deal efficiently with polygonal objects. This factor, together with recent developments in cheap rendering hardware, has secured the supreme position of the polygon mesh structure. This has resulted in a situation where it is more efficient - as far as rendering is concerned - to represent a shape with many simple elements (polygons) than to represent it with far fewer and more accurate but more complicated elements such as bi-cubic parametric patches.

There are certain practical difficulties with polygon meshes. The most important problem is accuracy. The accuracy of the model, or the difference between the faceted representation and the curved surface of the object, is usually arbitrary. The only way to increase the accuracy of the model is to increase the number of polygons at the expense of an expansion in the data. The optimal number of polygons depends on the trade-off between the accuracy and the rendering cost of the model. As far as final image quality is concerned, the size of individual polygons should ideally depend on local spatial curvature. Where the curvature changes rapidly, more polygons are required per unit area of the surface. Another problem is the manipulation of a polygon mesh. Polygon meshes do not allow simple shape manipulation. Moving mesh vertices immediately disrupts the ‘polygonal resolution’ where a shape has been converted into polygons with some degree of accuracy that is related to the local curvature of the surface being represented.

A significant problem that occurs in many areas in computer graphics is the scale problem. With polygonal representation this means that we cannot afford to render all the polygons in a model if the viewing distance and polygonal resolution are such that many polygons project onto a screen area that contains a single or only a few pixels. On the other hand the output of the rendering can look inaccurate if only a few polygons

project onto a big screen area. This problem occurs with complex and detailed virtual reality applications. The number of polygons in a complex scene is known as Level of Detail (LOD). An obvious solution is to have a hierarchy of models and use the one appropriate to projected screen area [SZL92, Hop96]. There are two problems with this. Firstly, the user can see an annoying visual disturbance as the details blink on and off from one resolution level to another. The other problem is how to generate the hierarchy and to decide how many levels it should contain.

2.1.2 Constructive solid geometry (CSG)

CSG [Wat00] is a convenient representation for architectural models. It has been used where the models focus on the whole structure of the building rather than on small details [DTM96]. This representation is used for CAD applications and also for reconstructed 3D models (*e.g.* the FACADE project [SA00] or Braun in [BKL+95]). CSG is very suitable for representing architecture and man-made objects in general. These kinds of objects tend to consist of many regular-shaped components. CSG has arisen out of the realisation that many man-made objects can be represented by combinations of elementary shapes or geometric primitives. The 'logic of the shape' in this representation is how the final shape can be made or represented as a combination of primitive shapes. The designer builds up a shape by using 3D building blocks and a selection of ways in which they can be combined. Such a representation provides very easy and intuitive shape control. Because of the ease of control, this form is widely used for CAD applications. The user can manipulate it easily. It is a high-level representation that functions both as a shape representation and a record of how it was built up.

The CSG approach is very much a user-oriented representation and requires special rendering techniques or the conversion to a polygon mesh model prior to rendering. CSG uses a volumetric representation - shape is represented by elementary volumes or primitives. This contrasts with the polygon mesh which represents shape using surfaces.

The motivation for this type of representation is to facilitate an interactive mode for solid modelling. The idea is that objects are usually parts that will eventually be manufactured by casting, machining or extruding and they can be built up in a

CAD program by using the equivalent (abstract) operations combining simple elementary objects called geometric primitives. These primitives are combined using (three-dimensional) Boolean set operators or linear transformations. An object representation is stored as an attributed tree. The leaves contain simple primitives and the nodes store operators or linear transformations. The object is built up by adding primitives and causing them to combine with existing primitives. Shape can be added to and (in order to make holes) subtracted from the current shape. For example, increasing the diameter of a hole through a rectangular solid requires a trivial alteration - the radius of the cylinder primitive defining the hole is simply increased. This contrasts with the polygon mesh representation where the same operation is distinctly non-trivial.

Although there are substantial advantages in the CSG representation, it also suffers from drawbacks. The most important problem is the computation time required to produce a rendered image of the model.

2.1.3 Boundary representation

The boundary representation of a model uses surfaces to describe bounding surfaces in the scene. The surfaces are represented by complicated data structures giving information about surface shapes and positions and how they are joined together. The surfaces are either described by its positions in 3D space as given by its (x,y,z) coordinates or parametric (or algebraic) descriptions. This representation has been used for architectural reconstruction where planar models are required [LCZ99, SM99, BZ00, WBS01]. Planes are described with the surface normal and the distance to the origin. Furthermore, such a representation does appear in ray tracing where bounding spheres are used frequently.

2.1.4 Implicit representation

The implicit representation of a model uses implicit functions to describe the surfaces of the objects in the scene [Wat00]. It can loosely be described as objects formed by mathematically defining a surface that is influenced by a collection of underlying primitives such as spheres or planes. Similarly to CSG, representation with this tech-

nique is of limited usefulness because there is a limited number of objects that can be represented in this way (*e.g.* planes, cylinders and spheres). In fact, this representation can be used to describe CSG and bounding objects. The implicit representation is not very suitable for representing real scenes containing many details. Also, it is an inconvenient form as far as rendering is concerned.

Implicit representations have their main use in shape-changing animation. A significant advantage of implicit functions in an animation context is the ease of collision detection that results from an easy inside-outside function (*i.e.* it is easy to distinguish between points inside and outside an implicit object). The problem with using implicit functions in animation is that there is not a good intuitive link between moving the centre of an object and the deformation that ensues because of this. Of course, this general problem is suffered by all modelling techniques where the geometry definition and the deformation method are one and the same. Another problem is that unwanted blending and unwanted separation can occur when the centres of objects are moved with respect to each other and the same blending method is used.

2.2 Geometric constraints

Often, automatically acquired models of well-known objects like industrial parts appear unnatural due to errors in the reconstructed model geometry, such as surface ripples on a supposedly flat plane or jitter on supposedly straight edges. Other unwanted artefacts include holes in the model due to surface occlusions and imprecisely aligned surfaces. In the case of industrial parts, however, it is normally important to have a precise model of the object for reverse engineering purposes. Constraint-based modelling has often been used for increasing the precision of models of a large variety of objects. Objects modelled range from small industrial parts [WFAR99b] to human bodies [LC01] or landscapes [Toe96]. For industrial parts, for example, geometric constraints are applied to model features like planes or cylinders. A cylinder can, for example, be constrained in its size, shape or location.

Often the application of constraints leads to over-constrained systems. One can analyse the constraint system to find redundancies in the system [LHS01]. The use of

a minimal set of constraints leads to a computationally faster, more accurate and more stable constraint application.

2.2.1 Industrial parts

It is sometimes necessary to reverse-engineer existing industrial parts, if original specifications or computer models of the object are lost. Reverse engineering is motivated by the need to produce a copy of an industrial object or the desire to analyse and modify a part in order to construct a new improved one. Application of constraints during reconstruction is useful, because manufactured parts are generally designed with intended geometric properties of object features and relationships between object features. Constraints on industrial parts can be fairly specific [WFAR99b, RFWA00]. A certain angle between planes, the shape of a surface (*i.e.* cylinder) and the distance between planes are examples of constraints used here.

Common quadric surfaces such as cylinders and cones are found in many industrial parts. A reliable estimation of the quadric surface might be difficult, if only a partial area of the surface is visible in the data or a part of the surface is occluded by another surface. Combining this with measurement noise and segmentation errors makes it difficult to fit the surface accurately to the data points. The extraction of quadric surfaces from range data can be greatly improved by using constraints [WFAR99a]. The poorness of the information embodied in the quadric surfaces is compensated for by extra knowledge about the surface such as the surface type and relationship to other nearby surfaces. In [WFAR99b] a general incremental framework for reconstruction of manufactured parts by application of multiple coupled non-linear constraints is described. Constraints are applied to points, lines, planes and quadric surfaces. They are defined in a special constraint definition file, which includes the exact location of the model features. The set of constraints associated with a given object can be divided into two categories. The first one covers the geometric properties which depend on the shape of the surface. The second category defines the geometric and topological relationships between the different object features. The parameterised model is optimised with the Levenberg-Marquardt algorithm. The parameters of the objects are varied to satisfy the constraints and to fit to the data, resulting in a trade-off between minimisation of the

shape fitting error and the constraint tolerances. Inconsistent constraints are detected by checking if constraints are not satisfied at the end of the optimisation.

Another approach for reliable reconstruction of industrial parts from poor quality range data is presented in [RFWA00]. Surface discontinuities are extracted by applying a local gradient operator. Then, parametrised models for drilled holes and slots are fitted to the discontinuities, using a RANSAC algorithm [FB81]. Several constraints are defined for the models. Examples of constraints are that slots have straight edges; all slots have the same width and slots are all lying in the same direction or holes are circular and the radii are known. The parameterised models of the industrial part are optimised by a genetic algorithm (Genocop III [Mic96]) such that the data fits well to the models and satisfies the shape and relationship constraints. The genetic algorithm tries to minimise the evaluation function which includes the the geometric distance to the theoretical primitive for each point. The shape and relationship constraints are formulated as equation and inequality linear and nonlinear constraints on the models parameters.

Another approach to applying constraints in the process of reverse engineering is presented in [LMM02]. Firstly, regularities [LMMM01] and also symmetries [MLMM01] are extracted. Regularities are extracted by seeking for similarities between object parameters. The application of the constraints, in contrast to the above techniques [WFAR99b, RFWA00], does not refer to the data points during model optimisation and so avoids the computational expense of constrained model fitting. Parameterised models of industrial parts consisting of geometric objects such as planes, spheres, cones, cylinders and toruses are used. The geometric objects are described by a set of appropriate directional, positional, length and angular parameters. Regularities are relationships between two geometric objects. Constraints are created to enforce the regularities. However, not all constraints may agree with each other. Therefore, a subset of the constraints is obtained by removing inconsistent constraints as follows. First, easily-detected inconsistencies between constraints of the same kind on the same geometric object are removed. A quasi-Newton method is then used to minimise the least squares error of the constraints. To detect numerically inconsistent constraints, the constraints are added one by one. When a constraint is added and its addition produces

an invalid solution, it is removed again and discarded as inconsistent.

The work described above shows how to apply complicated constraints to improve the precision of reverse engineering models of manufactured parts. Geometric constraints are applied to the parameters of the models. However, the selection of the constraints and its location of application is either manual [RFWA00, WFAR99b] or the 3D model is over-constrained [LMM02] and a set of consistent constraints must be found in a time-consuming process.

2.2.2 Occluded surfaces

Scenes are often only reconstructed partially. Holes in the model result from occluded surfaces in the sensor data. Typically, rather few images of the scene are used to perform the reconstruction. However, even in moderately complex scenes many images are necessary to see all surfaces in the scene. In areas like reverse engineering of manufactured parts it is not sufficient to reconstruct an incomplete 3D model. Precise and complete models are needed for further design and analysis. In an area like architectural reconstruction, incomplete models might look unnatural to the beholder. For example, a tree might occlude a part of a house façade and would result in a hole in the 3D model of the façade.

To minimise occlusions, best next view algorithms have been proposed [Pit96, RAS97, SF99]. These find the areas with the biggest occlusions and calculate the scanner viewpoint that would fill the the holes behind the occluding objects best. Another way of filling-in the gaps is to use prior knowledge. Man-made scenes normally consist of objects with continuous surfaces. Partially occluded surfaces like planes and cylinders can be fully reconstructed by detecting the occlusions and placing points onto the hypersurfaces [SDF01]. Additionally, occluded edges, corners and the surrounding surfaces can also be reconstructed [CLF02]. Straight edges are commonly found on man-made objects. Surrounding edges are extended. Then, the surrounding surfaces are extended to fill the space between the extended edges.

2.2.3 Computer aided design (CAD)

CAD systems are used to design all kinds of structured objects. CAD models typically consist of many geometrical primitives such as planes or cylinders with certain geometrical properties. Geometrical constraints (in CAD terminology called variational constraints) can be used at the stage of creating and manipulating CAD objects to ensure certain desired geometrical shapes and relationships between features, such as a certain angle or distance between two planes. By changing the constraints the CAD model can be manipulated using a small number of meaningful parameters rather than a large number of coordinates. For example, an architect might want to make certain storeys of his skyscraper design taller. Although conceptually simple, this operation might force the architect to change thousands of coordinates for various building components, a work-intensive and error-prone task. As a result of this difficulty, complex CAD models are often not created until the design is nearly complete. Geometric constraints can be used here. They automatically maintain desired spatial relationships between the features.

Constraints are separated into shape constraints (*e.g.* a hole has a certain diameter) [SAK90] and location constraints (*e.g.* a hole is at a certain location on a plane) [Tur90]. Manipulating the geometry of a model often requires interaction between shape- and location-constraints. The two constraint kinds can be coupled. In [Mar95] a hierarchical constraint approach is described which is suitable for geometrical structures including buildings. Operations on the model typically involve parameters influencing both the shape and location of many parts. A hierarchical order on the constraints reduces the effort required to modify complex 3D models.

2.2.4 Landscape

Landscape modelling is used for applications such as landscape planning, environmental monitoring and flight and driving simulations. The common techniques for recovering an elevation map from overlapping aerial images are stereo based. To obtain 3D models for efficient visualisation the elevation map is approximated by a triangular mesh in 3D space. Details of the landscape are modelled by projecting the aerial

images onto the mesh surface. However, the resulting models often don't have the required precision when viewed up-close. Subsequently, edges of the forests in the images do not correspond correctly with the height step in the 3D data and roads do not run continuously. These problems are caused by mesh approximations that do not correspond with the object boundaries. Furthermore, the sensor data includes noise; and important objects like roads are often partly occluded. Prior knowledge about object geometry can be used to improve the landscape model [Toe96]. First, the scene is interpreted. The aerial images are segmented into various regions, such as forest, grassland, rivers and roads. This knowledge, applied as constraints to the triangulated mesh, forces a height step between forests and grassland or roads. Furthermore, roads and rivers are approximated by a separate mesh to ensure a continuous course. So, occluded parts of the roads and rivers lost due to image resolution are inserted to obtain a consistent model.

2.2.5 Human models

Dynamic 3D reconstruction of articulated structure such as humans in motion normally use many cameras in a controlled environment in order to maximise the stability of reconstruction and handle the problem of self-occlusion. Reconstruction of the 3D motion of a single person in a real world environment with a single or few cameras is much more challenging. Human bodies have traditionally been modelled as "rigid link" articulated structures [LC01]. Knees, shoulders and elbows are rotational joints connected by (generally) rigid links. The lengths of body segments between rotational joints therefore remain constant over time. These properties (*i.e.* constraints) are exploited for both camera calibration and dynamic 3D reconstruction of a human person.

2.3 Architectural constraints

The highly-constrained structure of architecture has been the subject of much previous research in scene modelling/reconstruction which has aimed to create realistic architectural 3D models from various input sources. In this section we investigate which kinds of constraints have been used to support the creation of such models. Application

of constraints to architectural scenes is very suitable, since geometry of these scenes is typically very structured while at the same time they are one of the most common types of environment one would wish to model. Constraints on architecture are typically more generic, since architecture is done subject to certain accepted rules. It is unlike constraints on industrial parts which change from object to object and are very specific, although more specific constraints, like a certain angle between two walls, may still be applicable [Gra97].

2.3.1 3D Modelling

A major goal in 3D computer graphics is the creation of models that appear just as real as the world we are living in. But while 3D graphics techniques have made incredible advances during the last years, it has remained difficult and work-intensive to create realistic 3D models. Debevec [DTM96] presented an easy-to-use interactive software environment to model existing architectural scenes from multiple digital photographs. The environment uses architectural constraints to help the user create accurate models of the scene. The user builds a 3D model (CSG) of the scene by selecting geometrical primitives such as blocks and arches. Edges of the primitives are linked to edges in the images manually. The location and the shape of the primitive is then computed so that the model agrees with the photographed geometry. Basic constraints such as parallelism of primitives and edges are used to align them. The technique also allows the exploitation of architectural symmetries, since the shapes of repeated structures are only computed once. Additionally, to make the resulting 3D model look more convincing, view-dependent texture mapping is used. Each pixel from the digital photos which view the surface point contributes to the rendered surface point by weighting dependent on how closely each image's view of the pixel is aligned with the desired view. The choice of a CSG representation simplifies the modelling and leads to very structured models. This representation greatly reduces the dimensionality of finding the model parameters and leads to more robust and convincing models.

This software environment produces impressive results. However, an experienced user must control the reconstruction process. Furthermore, only limited buildings consisting of very basic shapes can be modelled without modelling any details.

2.3.2 Augmented reality

The goal of augmented reality is to insert virtual objects into real video sequences in a way that they appear as part of the real video. This technique has been used for creation of many modern movies. However, one problem here is the accuracy of the augmentation. It must not drift or wobble in the scene. A human observer can easily spot these problems, because of the human visual system's high sensitivity to perceptual cues such as motion parallax and parallelism. In [SFZ99] research is presented which inserts additional buildings or a changed façade into a video sequence of a city. The positions of the inserted objects are adjusted by maximum- likelihood estimation using constraints such as parallelism and tangency.

2.3.3 3D Reconstruction

As 3D graphics becomes an everyday feature of common desktop PCs, there is a high demand for the quick and cheap generation of realistic 3D models. The automatic reconstruction of 3D scenes has been actively pursued by computer vision researchers in recent years [FRL+98, DTC00, WZ02b]. However, algorithms used for reconstruction are very sensitive to errors in the source data caused by noise processes in the sensing. The common way to reduce the noise is surface smoothing [Cai89, TF92]. For range data it is also possible to use an outlier removal algorithm [UW96]. In this section we discuss the use of constraints to improve the structural quality of reconstructed models.

By using constraint-based reconstruction techniques, model inaccuracy, which occurs in the form of noise, is reduced. Certain relationships (*e.g.* two planes are parallel to each other) and shapes (*e.g.* a cylinder has a certain radius or the planar surface is flat) are enforced on the features in the reconstructed model. Furthermore, constraint-based techniques require fewer photographs and even allow single-view reconstruction [LCZ99]. The reconstruction of hidden surfaces that are not visible in the image but can be inferred from knowledge of the constraints is possible as well [RC00, DTC02].

In this section, we look into how constraints have been used to improve the reconstruction from intensity images; aerial images as a special case of intensity images; and range data. Then, we discuss techniques that exploit constraints to improve recon-

structed triangulated 3D models.

2.3.3.1 Intensity images

Using photographs as the underlying scene primitives provides an inexpensive and realistic environment representation. Several methods, together known as Shape-From-X, have been used to compute the geometry of the sensed scene. However, it is not always easy to obtain the structure of the scene, since intensity images only represent the 3D structure of the scene indirectly. The common technique for calculating the depth of the scene is stereophotogrammetry. The 3D structure of a scene is estimated from two or more intensity images taken from different viewpoints. For this approach it is important to determine which point in one image corresponds to which point in the other image ("The Correspondence Problem"). The accuracy of the resulting model depends directly on the accuracy of the feature correspondences. Constraint-based modelling techniques allow improved accuracy and single-view reconstruction by plane rectification. Besides reconstruction, scene constraints have also been widely used for camera calibration [SM99, WBS01, BPM01].

The reconstruction of a building from a single image of the scene might be necessary in cases where the building was destroyed and only archive images are available [BBV01]. Naturally, there are limits on the kinds of scenes it is possible to reconstruct and on the degree of completeness of reconstruction achievable. To perform reconstruction from a single view, geometric information determined from the scene is necessary. Planar models of buildings are derived from one or two images by exploiting geometric relationships that are common in architectural scenes, such as parallelism and orthogonality of planes. The technique presented in [LCZ99] uses vanishing lines for rectification of the planes and determination of their orientation relative to the camera. The minimal information necessary is the vanishing line of a reference plane and a vanishing point for a direction not parallel to the plane [CRZ00]. The reconstruction is performed without knowledge about the camera's internal parameters nor of the explicit relationship between camera and world (*i.e.* the camera pose). Parallelepipedic structures, which are a natural building block of realistic models for man-made scenes, are used in [WBS01] to perform the camera calibration and 3D reconstruction from

one image.

In [SM99] the user provides information about coplanar points and perpendicular and parallel edges and planes to perform the reconstruction of 3D points and planes from a single image. Perpendicularity constraints are also used to calibrate the image. Together with parallelism constraints they provide the vanishing point geometry of the scene which forms the skeleton of the 3D reconstruction. Coplanarity constraints are used to complete the reconstruction by estimating the scene planes and points. However, several points may lie on more than on plane. This means that, due to measurement noise, it is difficult to obtain a 3D model that satisfies all constraints exactly. Their approach [SM99] uses a direct least-squares solution for reconstructing a subset of object planes and points, minimising the sum of squared distances between planes and points.

Reconstruction of 3D models from a single image can only produce limited representations of the scene as the scene can only be seen from a specific viewpoint. The use of multiple images, which more sufficiently cover the scene, leads to a much fuller 3D model with fewer occlusions. Reconstruction from multiple images does not necessarily require constraints to create a 3D model [Pol99]. However, several approaches use them to simplify the reconstruction process and to improve the quality of the model. Faugeras [FRL+98] showed how to create a realistic textured model of a scene from a sequence of images, without any prior knowledge either about the parameters of the camera or about camera motion. The camera is calibrated and the 3D structure of the scene is reconstructed by using scene constraints such as parallel lines, ratios of lengths and angles. 3D lines on the same plane can easily be grouped together with the prior knowledge that they are coplanar [BZ00]. A set of planes constrained by the 3D lines is hypothesised in space and plausible plane hypotheses are identified by checking similarity over multiple images of the same scene. Additional lines, which were not extracted during feature detection, are created in the the reconstructed model by searching for plane intersections.

Much work concentrates on the reconstruction of the exteriors of buildings. An interactive technique is described in [RC00]. A few user-provided feature correspondences are used to estimate the scene structure. The strategy exploits parallelism and

orthogonality constraints on line directions and plane normals. An entirely automatic approach to derive architectural 3D models from multiple images is shown in [DTC00]. From an initial reconstruction obtained from corner features, a model based on planes is created and then optimised by exploiting the prior knowledge that walls are likely to intersect at about 90 degrees. Additionally, common architectural features such as rectangles and arches are refined by using parameterised shape models. This technique has been extended by incorporating a probabilistic framework for finding architectural features such as windows, doors and buttresses in the scene [DTRC01]. The reconstructed 3D model is built-up from parameterised parts corresponding to the features found in the scene. Because the scene has been interpreted, the shape and texture of the resulting 3D model can be reconstructed more reliably where evidence from the images is weak. Furthermore, by taking the interpretation into account, the model can be enhanced by application of reflective texture to windows .

A different kind of constraint is used in [WZ02b]. The principal directions of the scene (see section 3.1.2) are used to constrain the reconstruction of architectural scenes from multiple images. Architectural scenes typically contain planes orientated in three dominant directions which are perpendicular to each other, for example the front and side walls of a building and the ground plane. A coarse piecewise-planar model of the principal scene planes and their delineations is reconstructed. This method has been extended by fitting shape models to windows [WZ02a]. It recognises these models in the images by using a Bayesian framework (similar to the method described in [DTRC01]).

Constrained 3D reconstruction of architecture can be closely coupled with an interpretation of the architectural structure. In [WG96], a knowledge-based approach for reconstructing the wireframe model of a building has been proposed. The building structure is estimated from pairs of stereo images. Regions and contours of the scene are extracted from the 2D and 3D data. These extracted features are assigned semantic meanings in the interpretation step. The house knowledge is formulated in a semantic network with three layers. The top layer describes high level features such as walls, the middle layer represents 3D features such as triangles and the bottom layer includes the 2D image features like 2D triangles and regions. The features are matched against

the house model with a hypothesis-driven approach. This approach tries to verify the matches in the image data. The scene interpretation is used to select architectural constraints from the knowledge base. The extracted features together with the selected constraints are the input for the surface reconstruction process. The constraints used here describe a relationship either between parts of the 3D model (*e.g.* orientation of walls to each other) or between a part of the 3D model and the 2D image. The application compensates for noisy and sparse depth maps.

In [BKL+95], an approach is presented which exploits constraints to improve CSG models (see section 2.1.2). Constraints are defined on the parameters of the primitives in the model and the attributes of the relationships between primitives. Constraints on primitives define restrictions on the primitive parameter set. Constraints on relationships define restrictions on the attributes of the relationships between the primitives. These constraints include probabilities to specify uncertainties in the measurements.

Much of the research described above exploits constraints that are described in the images. The scene, however, may also be a rich source of constraints that can be exploited. A rich source of constraints is a detailed map of the scene. Such maps are often available for urban scenes. Maps are locally much less accurate than images (*e.g.* on the scale of a single building), but they provide a strong constraint on absolute geometry (*e.g.* orientation of buildings to each other). Camera calibration can be performed with a single image and a map of the scene [BPM01]. The use of maps enables the accurate reconstruction of large-scale models. [RC02] shows how to use a map together with many uncalibrated images to reconstruct a large architectural scene including several buildings.

As shown, much work has been done on reconstruction of architectural scenes from intensity images. The main attraction is that a relatively cheap digital camera is sufficient as a sensor device. Architectural constraints make the reconstruction problem much easier to solve. Unfortunately, many reconstructed models do not show detailed structural information. Often only planar models are reconstructed. Furthermore, the size of the 3D model reconstructed is often very limited. However, the use of map-constraints enabled recent work to reconstruct substantially larger models. In this area of research constraints have been mainly used to compute the reconstruction and only

secondarily to archive better results.

2.3.3.2 Aerial imagery

The use of aerial imagery (taken from an aeroplane or a satellite) is a special case of architectural reconstruction. Over the last few years extraction of cartographic features from aerial images has become a subject of intensive research. The main application here is to provide an automatic cartographic process to create and update maps, particularly in urban areas. Stereophotogrammetry is the basis of most methods. Man-made features, such as road/railway-networks and buildings have been given special attention [CJC98, Toe96].

Cord *et al.* [CJC98] combine 2D monocular and 3D stereo analysis to reconstruct urban environments from mid- and high-resolution aerial images. Monocular analysis is used to detect edge features, extract contours and identify regions. Regions are characterised by four parameters: region size, average grey level, quality rate of the approximation by a parallelogram and compactness. They are classified by a neural network classifier and a constraint propagation system as: lawn, grove, building, tree, shadow and vertical face. Stereo analysis of an image pair is then used to support feature grouping, building recognition and building height reconstruction. The reconstruction process exploits the fact that man-made objects possess a large amount of geometric regularity (*e.g.* flat rectilinear roofs). Simple constraints that describe the nature and shape of a normal house roof are used: roof surfaces are planar and buildings are supposed to have linear edges.

Ascender I [CHR+96] is a system for building reconstruction from aerial imagery. It uses 2D image features and grouping operators to detect house boundaries. Known geometric constraints are applied to increase the efficiency and reliability of the reconstruction process. Parametric planar, peaked or curved models are fitted to the roofs. The visual realism of the texture is improved by extracting windows and doors on wall surfaces and rendering them dark and opaque. Ascender II shows how semantic interpretation can improve the reconstruction of buildings from aerial imagery [MJHR99]. Image regions are classified as house walls and roofs with a Bayesian network.

Another method for semantic scene interpretation of roof models is presented in

[SMG02]. First, 3D line segments are extracted using feature-based multiview correspondence analysis. The 3D lines are grouped into planes. Planes are linked together with constraints to form roofs. These estimated roofs are semantically interpreted. The boundaries of the roof are labelled as ridge, gutter, gable, convex or concave using a linear discriminant analysis. Following this, small gaps between planar patches are closed and missing roof parts are reconstructed by using the labelling.

Additionally, colour can be an important clue for the reconstruction [Hen98]. Colour is very helpful both for image analysis and subsequent surface extraction and for easier building recognition. Regions belonging to the same object are likely to have similar colour properties. This knowledge reduces the computational complexity of finding and segmenting house roofs in the data.

2.3.3.3 Range data

Intensity images are of limited use in terms of estimation of surfaces. Pixel values are related to surface geometry only indirectly. Range images encode the position of surfaces directly. Each pixel of a range image expresses the distance between a known reference frame and a visible point in the scene (2.5D). Therefore, a range image reproduces the spatial structure of a scene. Furthermore, several fused range images taken from different viewpoints represent the 3D structure of a scene.

Unlike in the previous section, constraints are not employed to simplify the surface reconstruction. Because of the nature of range data the surface is already explicitly encoded in the data. With range data, architectural constraints are used solely to improve the structural quality of the 3D data.

Wolfart *et al.* have proposed a hybrid approach for reconstruction of triangulated 3D models from range data [WSN+99] which combines geometric surface extraction and robust triangulation. First, they triangulate the surface from the range data. This results in an initial 3D model with noise visible especially on large planar surfaces. Typically, in an architectural scene, a large part of the scene area consists of planes. To remove the noise on the plane, these planar patches are identified by fitting planes to the data. With the knowledge of the planes, coplanar constraints are applied to the data. The result is that large surface areas appear perfectly smooth. At the same time,

the approach preserves a high level of detail for objects not on planes like small objects on tables. Beside enforcing surface flatness, constraints can also be applied to edges to enforce straightness [MF02]. For this, fold edges are detected in the range data and the triangulation of the scene is constrained by placing seed triangles onto the edges. This prevents triangles from cutting under the edges.

Architectural constraints have been applied to more specific architectural structures as well. In [FF02] constraints are used to improve the shape of reconstructed models of windows and doorways from range data. The objects constrained here are represented by a parameterised model. The parameters of the model are optimised with a genetic algorithm (GA), similar to the method described in [RFWA00], such that the data fits very well to the model and simultaneously the constraints are satisfied. The chromosome is the data vector which is optimised by the GA. It consists here of the model parameters. The evaluation function is based on the closest distance of each data point to the closest surface of the 3D model. Therefore, the evaluation function corresponds to the degree that the 3D data fits to the parameterised model. All constraints used are defined within the genetic algorithm. Their satisfaction depends on the model parameters in the chromosome.

Besides reconstruction, constraints can also be applied to the problem of automatically matching 3D and 2D data from range and image data sets [SA01]. Parallel and orthogonality constraints are utilised to extract the 3D rectangular structure from the range data and 2D rectangular structure from the intensity data. RANSAC [FB81] is then used to to obtain an optimal match between the two rectangle feature sets.

Wolfart and Faber [WSN+99, FF02] apply constraints to 3D data to improve the reconstruction of 3D models. They concentrate on single features, however. Wolfart applies constraints to planar surface patches, but without taking the orientations of the surfaces into account. Faber constrains more specific features. He fits parameterised models to windows and doorways.

2.3.3.4 3D mesh

Similar to the range data described in the previous section, 3D meshes inherently represent the 3D structure of the scene. A 3D mesh is a commonly used representation for

modelling architecture (see section 2.1.1). Reconstructed 3D meshes are likely to include small irregularities in the form of small surface ripples, because of measurement noise. This is unlike reconstructed CSG models where each model consists of primitives with smooth surfaces. With 3D meshes, constraints can be used to compensate for noise and so to increase the quality of the reconstruction.

A user interactive approach for quality enhancement in 3D modelling is described in [RDSG01]. A user has to select semi-automatically a plane or an edge either on the triangulated model alone or combined with registered video. Specific geometric algorithms are then applied to correct data imprecision. Edges are straightened, planes are flattened or peaks on planes are removed. The optimal plane or edge parameters are computed by using orthogonal regression. The mesh is modified to accommodate the correction if necessary.

One problem in 3D reconstruction based on laser range data is the limited spatial resolution of the depth measurements. This results in jitter on roof and other crease edges. Information from intensity images and 3D data can be used to straighten edges in architectural triangulated 3D models automatically [DSGV01]. The intensity information is used to help find straight lines in the scene. The corresponding lines in the mesh are selected. The line's parameters are refined by computing a linear regression fit. The resulting 3D lines are straight and the best fit to the original 3D data. The refined parameters are used to project the line vertices onto their lines.

Some work has been presented to improve 3D reconstruction of triangular meshes. [RDSG01] showed a comprehensive method for flattening planes and straightening edges. Unfortunately, this method is mostly user interactive. Dias *et al.* [DSGV01] used an automatic technique. However, they only constrain edges, but no surfaces.

2.4 Summary

This chapter has discussed the use of constraints for modelling man-made objects and in particular buildings. Furthermore, an overview of commonly-used 3D representation has been given.

Several representations for 3D models are commonly used. Planar models are fairly

limited in use, because they only represent planes. CSG is a suitable representation for large models with regular structure. However, the most popular representation is the polygon mesh, because it is fairly easy to create automatically if a few 3D coordinates have been estimated and because it is computational very cheap to render. Large areas can be efficiently represented by large triangles. The representation can be used to model small details with smaller triangles as well. Triangular meshes are particularly suitable if only parts of buildings are reconstructed.

Several kinds of constraints for improving the reconstruction of 3D models have been presented. Geometric constraints applied for example to industrial parts can be very specific. They typically work on primitives such as planes or cylinders. Architectural constraints on the other hand are much more general. They have been widely used in 3D modelling, 3D reconstruction and camera calibration. Features used for architectural constraints are typically straight lines, large coplanar regions and the parallelism and orthogonality of lines or planes. Architectural scenes are typically very rich in these kinds of features.

Constraints have been applied for various reasons. They are exploited to increase the accuracy or appearance of the models, to simplify the creation of them or to reconstruct occluded regions of the scene. Beside reconstruction, they are also used for camera calibration and registration. Precise models are of special importance for reverse engineering of industrial parts. For reconstruction from intensity images, the use of constraints makes the problem of reconstruction easier [DTM96], fewer images are necessary to perform the reconstruction and even reconstruction from a single image is possible [LCZ99]. Improved appearance of models is essential for publicly-displayed models. They must meet certain expectations of human observers. Occlusions and geometrical distortions are particularly critical. Inaccuracies are recognised immediately and appear to be disturbing [SFZ99].

Techniques for application of constraints differ much. 3D reconstruction from intensity images typically incorporates the constraints tightly in the reconstruction process. Other work applies the constraints in a post-processing step.

Unfortunately, no single work currently applies constraints to improve the appearance of complete house models including walls, the ground plane and the roof both au-

tomatically and comprehensively. Some work concentrates only on house windows or roofs. Faber and Dick improve parameterised models of common windows and doorways reconstructed from range data [FF02] or intensity images [DTRC01]. Scholze improves the reconstruction of roof models by interpreting the model and refining it using the interpretation [SMG02]. Other work only constraints edges or planes on architectural models. Dias applies constraints to straighten lines in the 3D model which have been detected in the intensity image of the scene [DSGV01]. Wolfart [WSN+99] flattens walls by fitting planes to them. Furthermore, many approaches require more or less input from a user like in [RDSG01]. Research concerned with improving reverse engineering models of industrial parts often leaves it to the user to specify the constraints [RFWA00, WFAR99b].

So, we can see that no single approach so far has applied constraints automatically and in a comprehensive way to architectural models. The purpose of this thesis is to demonstrate that this is possible.

Chapter 3

Architecture

This chapter discusses principal architectural properties that we can exploit for 3D reconstruction of architecture. We introduce fundamental architectural concepts that define the structure of buildings. Constraints that we can use in architectural reconstruction are derived from these concepts.

3.1 Basic concepts

Architectural structure defines three-dimensional volumes of space. The structure consists of many small elements that together form a complete unit. The single elements relate with each other to create a vision of the whole. The relationships are important for the quality of space, light, sound and also social interaction. We show below basic concepts that define the structural geometry in architecture.

3.1.1 Geometry of making

The methods and materials used for constructing architecture can already define the structural form. A table or a desk tends to be rectangular if it is made of regular-shaped pieces of timber. Similarly, the materials we use to build a house impose or suggest the geometry. Walls if put together from rectangular bricks form naturally rectangular doors and windows. Figure 3.1 shows the façade of a brick building. The structure of the bricks can be easily seen. The border of the bricks matches nicely with the border

of the windows.

Furthermore, the structure of a building depends on the skill of the builders. Much more craftsmanship is required for building a round wall or a dome. Especially in the early days of architecture, buildings were made up with planar walls (*e.g.* ancient Greek architecture). Only the most prestigious buildings include them into their structure, because of the time-consuming and expensive nature of their construction.

Figure 3.1: A façade of a modern brick building. The bricks form naturally rectangular openings.

3.1.2 Linear elements

The majority of architectural structures consist of linear elements [Chi96]. Linear elements are columns, beams or corners of buildings. They define a volume of space. Linear elements like columns can be used to support the overhead plane. Moreover, they form a three-dimensional structural frame. Elements in this frame can be orientated in three principal directions that are orthogonal to each other. The structural frame created from linear elements can go through whole buildings like in many modern office buildings.

An example of linear structure in architecture are the medieval palaces in Japan, such as the Katsura Palace in Kyoto. It consists of linear columns and beams that form together a three-dimensional framework for architectural space. Linear elements can be found in many modern buildings too. Office buildings for example are very

structured and have much linear structure. An example is the Hypolux Bank Building in Luxembourg designed by Richard Meier. Its front has many linear elements in the form of vertical beams and canopies to protect the inside from the sun (see figure 3.2).

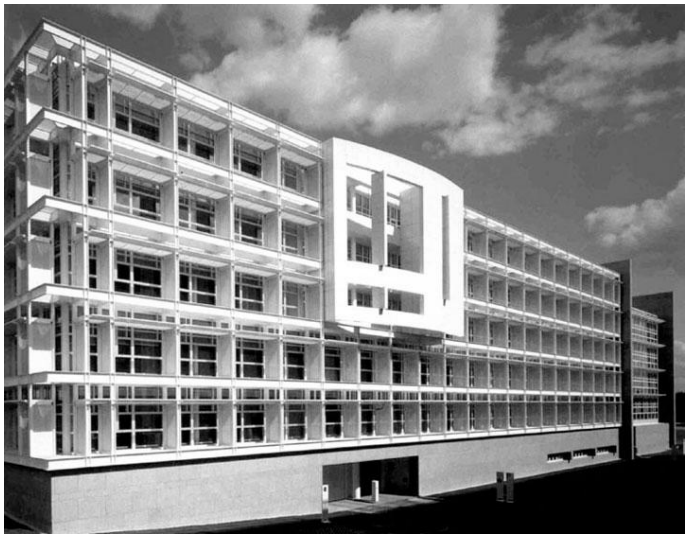

Figure 3.2: The Hypolux Bank Building by Richard Meier. The façade consists of much linear structure.

3.1.3 Planes

Planes together with linear elements define architectural space. Their spatial relationship to each other determines the quality of space they enclose. Architecture recognises three types of planes: the wall plane with its vertical orientation, the base plane that provides the foundation for the building form and the overhead plane (the ceiling or the roof) which shelters the interior space. In particular, walls are used as space-defining elements.

Parallel and orthogonal walls are the simplest, oldest and yet most enduring architectural structures. They are found in ancient architecture and they continue to be used in all kinds of architecture these days. The attraction of this most uncomplicated structure lies in its simplicity. It is very easy to build such walls on a base plane in comparison to building round walls or walls that meet at an angle of (say) 30 degrees.

Early examples of using parallel walls were found as far back as Ancient Greece. Parallel walls were used for ordinary houses as well as for temples. The openings were often orientated toward sacred symbols like mountains or stars. Similarly, Roman and Gothic churches have parallel side walls. The walls focus the place of the altar. Parallel walls (and orthogonal walls as well) have been used in house design too. Parallel structure easily allows repetitions to form terraced houses.

3.1.4 Six-directions-plus-centre

The Six-directions-plus-centre [Unw97] term derives from the human perception. The six directions relative to a human being (the centre) are the front, back, bottom side, top side and two sides to the right and left. The six directions are manifest on the surface of the earth. The sky is above, the earth below and the cardinal points define the sides. The cardinal points relate to the movement of the sun. In the northern hemisphere the sun rises in the east, gets its highest point in the south and sets in the west. We find and occupy places and relate ourselves to other places by using the six directions. The principle is that we build architecture like we perceive the world. Architectural design tries often to conform with the six directions and as such architectural structure responds to each of the directions. However, the six directions do not always appear in architectural structures - for example they do not appear in tepees or igloos.

The simplest example would be an ordinary cell with four walls, ceiling and floor. Another example is a Greek temple. The centre is the image of a god. Around the image is the base plane, roof and the four walls or lines of columns. But the building does not only relate this concept to its internal structure. The sidewalls of most four-sided buildings (good examples are temples or churches) on the surface of the earth point in some way, roughly or exactly, to the four cardinal points.

By its six sides a place (a room or a building) can set out a three-dimensional orthogonal framework as do the three principal directions for the structural frame (see section 3.1.2).

The simple use of the six-directions-plus-centre are basic, rudimentary and ubiquitously applied in many sorts of cultures.

3.2 Architectural constraints

The previous section described fundamental principles for architectural geometry. In this section, we will derive architectural constraints from these rules. Such, constraints are then used later to improve the structural quality for the reconstructed 3D models.

3.2.1 Planes

The plane is a very basic structural element in architecture. Planes have been used right from the beginning of architecture (see section 3.1.3) and indeed are used in the vast majority of buildings these days too.

3.2.1.1 Flatness

Although, ancient stone buildings might have rough walls and floors, the surfaces of modern buildings are rather smooth. However, it is important to distinguish between a plane and its decorations like pipes on the outside of buildings or images on interior walls. Only the actual plane should be flattened to preserve details of the model. Planes are flattened by applying coplanar constraints to them.

3.2.1.2 Alignment

There is a vast stock of orthogonal and parallel work in architecture as seen in sections 3.1.3 and 3.1.4. We can exploit such relationships by using orthogonal and parallel constraints.

3.2.2 Lines

Lines are the linear elements in architecture (see section 3.1.2). They occur at corners of buildings, openings like windows or doors and edges of the roof.

3.2.2.1 Straightness

Linear elements are typically straight. However, 3D reconstruction based on laser scanning distorts the depth discontinuity lines in the 3D model producing some jitter

on them. The problem is known as the “mixed point problem” (see figure 3.3). The measured position of a point on the line is a combination of the real point and a point further away on the other side of the line in the case where we have a jump edge. For roof edges one sees the same effect, although it is much weaker. To remove spatial jitter, a constraint which straightens the lines can be used.

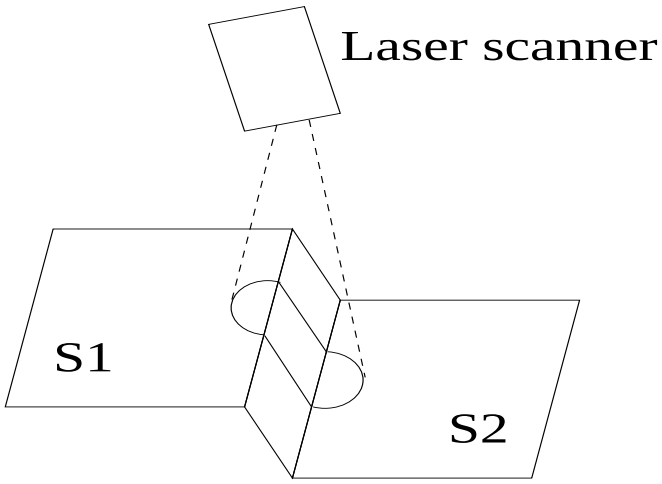

Figure 3.3: The mixed point problem: The laser scanner measures a combination of the distances for surfaces S1 and S2 at the depth discontinuity.

3.2.2.2 Alignment

As seen in section 3.1.2, linear elements are aligned to each other to form a structural frame. The orientation of the linear elements normally corresponds to the planes. Constraints can be used to enforce alignment of the lines in the 3D models. However, the number of linear elements can be much bigger than the number of planes in a model. Constraining a potentially very large number of lines leads to a very large number of constraints. For this reason, we group parallel lines in sets together and assign the same orientation to all of them. This means we do not need to use all these constraints. Three sets of parallel lines correspond to the three principal directions of the architecture. The orthogonal alignment of these three sets is enforced by three orthogonality constraints.

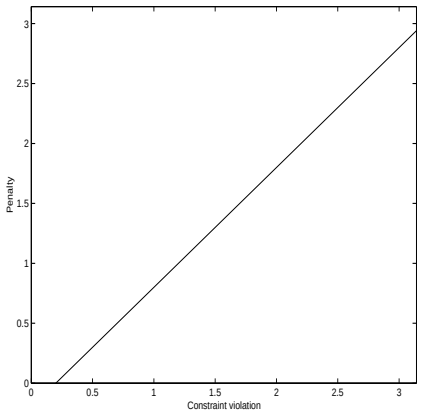

Figure 3.4: The constraints are modeled as penalty function. The penalty increases proportionally with the constraint violation. Violations below a certain tolerance (0.2 in the graph) are ignored.

3.2.3 Constraint satisfaction

Architectural scenes, like all man-made objects, are never 100 percent perfect. Walls are never perfectly aligned to each other. They will always be misaligned if only by a fraction of a degree. One has to decide to which degree to enforce the constraints.

The constraints are modelled as penalty functions. The penalty values correspond to the degree to which the constraints are satisfied. Perfect constraints do not have any penalty. As a constraint gets further and further violated the penalty value increases proportionally. We allow a certain tolerance however. The penalty values only start to rise after the constraint violation is above this tolerance (see figure 3.4). This approach preserves small misalignments in the data. The optimisation process thus has more freedom to adjust the model parameters.

3.3 Discussion

We reviewed fundamental architectural principles and derived architectural constraints from these principles. In particular, much architectural structure consists of planar surfaces and linear structure. However, one could also have done a survey on the

design of architecture to derive the constraints.

One might argue that the architectural concepts which have been presented here are too simplistic. Indeed, architecture is much more complex than the concepts suggest. Normally, buildings do not consist of basic cells put together with the same orientation. Nevertheless, orthogonal and parallel structure is found in many architectural works of many different kinds and characters.

Planes are flat and lines are straight. Does architectural structure really follow these rules? Certainly, interior or plastered walls and most lines do. However, in particular the façade of buildings is not always so smooth. It depends on the material which has been used. Brick walls have small gaps between the bricks. So do sandstone walls. Moreover, sandstone can be a bit uneven. Exposure to the elements may make walls uneven too. However, the structural disparity of walls is rather small compared with the whole architectural structure and is not always present in architectural 3D models. Often walls are modelled with rather few big triangles (triangular mesh) or just one big block like in CSG models.

Constraints could be applied to other surfaces such as cylindrical walls or domes (hemispheres) as well. Constraint reconstruction of industrial parts has used constraints on such surfaces for a long time [WFAR99b, RFWA00]. One could fit cylinders or hemispheres to those surfaces. Alternatively, more general techniques like peak removal [RDSG01] or surface smoothing could be applied here. However, surfaces such as cylinders and hemispheres are not very common in architecture.

Another issue is to which extent to apply the constraints. Do we want perfectly-aligned walls that differ less than (say) 0.01 degrees from a perfect orthogonal alignment? It depends whether we want to reconstruct a model of the architectural structure or a model of the real building. An architect considers two walls as parallel in his models even if the orientation differs by 0.5 degrees in the real building. The walls are always supposed to be parallel in his models. The real world however does not know perfect orientations. The orientation will always differ to some extent from the supposed ideal. We can reflect this small imperfection by constraining orientations only to a certain extent. However, then the question is how much inaccuracy was introduced into the building as it was being built and how much as it was scanned and triangulated.

We need to define the degree of misalignment we are prepared to tolerate here.

Chapter 4

Feature extraction

In order to apply constraints to regularities in the reconstructed model, we first have to extract geometric primitives. Feature detection here is an intermediate step. It is a necessary precursor to the extraction of regularities in chapter 5 and the application of the constraints in chapter 6 that improve the model quality.

For this work we are looking for primitives common in architecture. We reviewed architectural principles in the last chapter. According to our findings the most common primitives are planar surfaces and linear elements (*i.e.* 3D lines in our models). The process starts with detecting and locating these primitives in the data set. The primitives found are represented by a set of feature descriptors.

In this chapter, we give a short overview about 3D feature extraction. In the next section, we discuss how to extract planes and lines efficiently from triangular meshes. The triangular meshes we consider can be seen as the initial reconstruction that we then improve by using constraints (*e.g.* removing noise on the surfaces). Unlike a 3D point cloud, a triangular mesh is a surface representation. The 3D points (vertices) are connected by edges and triangles that determine the neighbourhood relationships between the vertices. We will exploit these relationships and the surface information in the mesh to improve the feature extraction. Texture data is not considered. We use the robust model estimation technique RANSAC [FB81] to find planes (similar to those given in [Bar01]) and lines in the model. RANSAC is able to deal with a gross amount of outliers in the data and noise on the surfaces - as is the case in our data. The RANSAC algorithm is presented in more detail in appendix A.

4.1 Overview

Feature extraction from 3D data is unlike feature extraction in 2D images. Extraction in images relies on colour or intensity information that gives important clues about features (*e.g.* lines). In contrast, a pure 3D point cloud contains no colour information, but three dimensional structural features (*e.g.* 3D lines) are inherent in its data. However, feature extraction within an irregular 3D point cloud is difficult, because it is not obvious which points are neighbours. 2D images are usually regularised in a grid. The neighbour relationships are obvious. 3D data could also be held in a grid (*e.g.* $2\frac{1}{2}$D data), but in general we have a point cloud with floating point accuracy in the coordinates. Computationally-efficient feature extraction methods for feature extraction in 2D images, like the Canny edge detector [Can86], do not have an equivalent in 3D, since they rely on the grid structure of a 2D image.

With pure 3D data, to start feature extraction it is important to establish neighbourhood relationships between the 3D points. It is then possible to estimate, for example, the local surface curvatures at each point with a local surface fitting algorithm. This provides an estimate of the principal curvatures, either Mean Gaussian curvatures [BJ86] or shape index curvature [KvD92]. Depending on the curvature, surface patches are classified as planes, free-form surfaces or edges. Complete surfaces are then found by an surface growing process that starts with initial seed regions [RFWA99]. Typically, large patches with the same curvature type are taken as seed regions. However, calculation of the derivatives for the curvature calculation is very sensitive to noise. Moreover, the curvature can only be calculated reliably if we have dense data. Range data is normally very dense. On the other hand, the triangulated models that we are dealing with often have a much smaller number of data points (vertices).

Calculation of the local curvature is an error prone process. Another way of extracting features such as cylindrical or planar surfaces from 3D data is to search the 3D data directly for instances of these features. The robust model estimator algorithm RANSAC [FB81] has been used to find planes [RFWA00] and lines [RF02] within 3D data. Random model hypotheses are created and the hypothesis which is supported by the most points is selected. This technique works well with dense 3D data, but falls short if features are only represented by a small number of points (*e.g.* planar surface

represented by large triangles and a small number of vertices).

However, here we do not work with pure 3D data. Neighbourhood relationships are embedded in the data, since points are connected by triangle edges. In [BV02] a segmentation process is described which divides the mesh into regions separated by sharp and smooth edges. Various tests are used to determine the properties of these regions. The new contribution in our work is the adaption of the RANSAC-based model estimation algorithm to extract features from triangular meshes. We evaluate the RANSAC hypotheses by calculating the hypothetical plane size and the line length, which is more meaningful than counting points. Our technique is able to extract features within sparse data.

At all stages, our data source is a triangulated mesh. It consists of vertices $V = \{(x,y,z) \,|\, x,y,z \in \mathbb{R}\}$ linked together by edges $E = \{(v_1,v_2) \,|\, v_1,v_2 \in V\}$ to form triangles $T = \{(v_1,v_2,v_3) \,|\, v_1,v_2,v_3 \in V\}$. The triangles in our data do not include any orientation. The size of them may be uniform or non-uniform. The mesh may consist of different meshes sharing vertices at the borders or separated by small gaps. It may include errors such as holes and peaks/edges sticking out of the surfaces. We assume that graver errors such as overlapping and intersecting triangles are not present in our meshes. Before starting the extraction the model is normalised to provide a scale-independent description and to allow process parameters of, and results on, different models to be more comparable. The model is scaled and translated to fit into a unit sphere at the origin.

4.2 Planes

Planes are one of the main structures in architecture. We start by extracting them from our triangulated models. This is done in three stages. First, we construct plane hypotheses. Next, these hypotheses are evaluated and the ones with the highest rating are selected. Finally, they are checked for discontinuities to ensure closely connected planes.

4.2.1 Plane hypothesis

The RANSAC plane algorithm hypothesises a number of random planes. For each hypothesis we need to draw a sample which contains the minimum number of data points required to construct a plane. Here the minimum sample needed for a plane consists of three points. The number of hypotheses depends on the likelihood of drawing a good sample from our data. Half the number of the hypotheses are constructed by selecting triples of random vertices $V_n = (v_{n,x}, v_{n,y}, v_{n,z})'$ from V. The other half of the hypotheses each use the three vertices of a single random triangle $T_1 = (v_1, v_2, v_3)$ from T. [MTN+02] concluded that one is more likely to find a good estimate which includes only "good" points if one selects points close together. However, selecting vertices that are too close together might give a bad estimate if the vertices are influenced by noise. Therefore, half of the hypotheses use vertices very close together and the other half use random vertices that are likely to be further away from each other.

For each hypothesis, the plane parameters are computed. First, the three vectors S_1, S_2 and S_3 between the three vertices are calculated as:

$$S_1 = V_1 - V_2 \tag{4.1}$$

$$S_2 = V_1 - V_3 \tag{4.2}$$

$$S_3 = V_2 - V_3 \tag{4.3}$$

The surface normal of the plane is then the normalised cross product between any two of the three vectors S_n.

$$N = \frac{S_n \times S_m}{||S_n \times S_m||} \qquad n \neq m \quad n,m \in 1..3 \tag{4.4}$$

Next, the distance to the origin D is computed as the dot product of the surface normal and any of the three vertices on the plane.

$$D = N \cdot V_n \qquad n \in 1..3 \tag{4.5}$$

To make the representation minimal, the surface normal is converted from the vector with x, y and z values into azimuth and elevation angles with respect to the reference

vector $(1,0,0)'$. Each plane is then represented by the surface normal with two floating point numbers and the distance to the origin with one floating point number.

4.2.2 Plane evaluation

Each plane hypothesis is evaluated. We identify the subset of the triangles T that lies on the hypothetical plane. First, the orientation of a potential triangle is taken into account. As a precondition, the triangle normal must vary less than a certain angle threshold from the plane normal. A triangle perpendicular to the plane most likely will not belong to it. A conservative threshold of 30 degrees works fine for this test. A triangle belongs to the plane if the triangle centroid is below a certain distance away from the plane. The individual vertices do not need to get checked, since the triangle normal is approximately the same as the plane normal. Thus, the three vertices lie about as close to the plane as the centroid.

The distance of a triangle centroid to the hypothetical plane is calculated by computing the absolute difference between the distance of the plane to the origin D and the dot product between the triangle centroid $C = (c_x, c_y, c_z)'$ and the unit plane normal N. Triangles that satisfy the following inequality belong to the hypothetical plane.

$$|C \cdot N - D| < tolerance \tag{4.6}$$

We need to determine how big the distance tolerance should be. In general specifying this threshold is part of the modelling process. It depends on how much one wants to flatten the model or how much structural details one wants to keep. Setting this threshold is relatively simple, since all models are normalised (*i.e.* mapped into a unit sphere at the origin). So, the process parameters applied to different models are easily comparable.

The size of a hypothetical plane is calculated by adding up the areas of the triangles that satisfy equation 4.6. The hypothesis that creates the largest plane is selected.

A common architectural structure typically includes several planar surfaces. To determine the exact number of planes is difficult. An expectation maximisation (EM) algorithm [NB97] could be used to select the number of planes and fit them. However, we only want to pick up the architectural planes and not the planes of, for example,

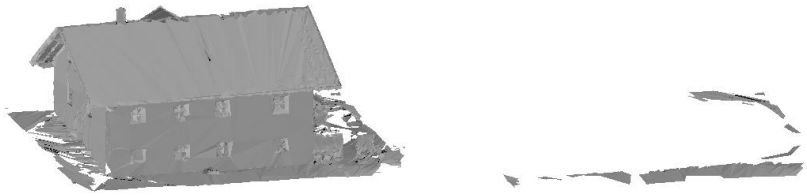

Figure 4.1: This is an example of a plane extraction where no connectivity constraint is used. On the left is a full triangulated 3D model of a Bavarian farmhouse. The model includes 4 walls, a roof with two planes, much details such as windows and doors and some of the surrounding ground. The plane extraction not only finds 6 planes for the walls and the roof, but also a plane which cuts partly through the surrounding ground (right). The extracted ground plane includes only a small part of the ground and moreover is fragmented.

furniture or doors. We are only interested in the largest planes. Therefore, we start by extracting the largest plane and repeat the algorithm on the remaining triangles until the size of the resulting plane falls under a certain threshold.

4.2.3 Connectivity constraint

The plane extraction method detects triangles lying on a plane well. However, it sometimes extracts a plane that consists of small disconnected patches distributed over the scene. Or, a perfectly valid plane such as a roof gets extracted, but also includes some triangles which are far away from it. These triangles lie by accident on the same plane, but belonging to another structure. Figure 4.1 shows a result where a non connected plane is extracted. An architectural plane (*e.g.* a wall) is not usually separated by a large gap. In a $2\frac{1}{2}$D or partly 3D model the middle part of a wall could be totally occluded by another structure and the wall consists of two planar patches. But, in a full 3D model disconnected planar structures are usually not part of one structure. Nonetheless, small gaps frequently occur due for example to the presence of pipes or decorations. Therefore, the planes are analysed by single linkage clustering [JD88] to ensure that the triangles of a plane are closely connected.

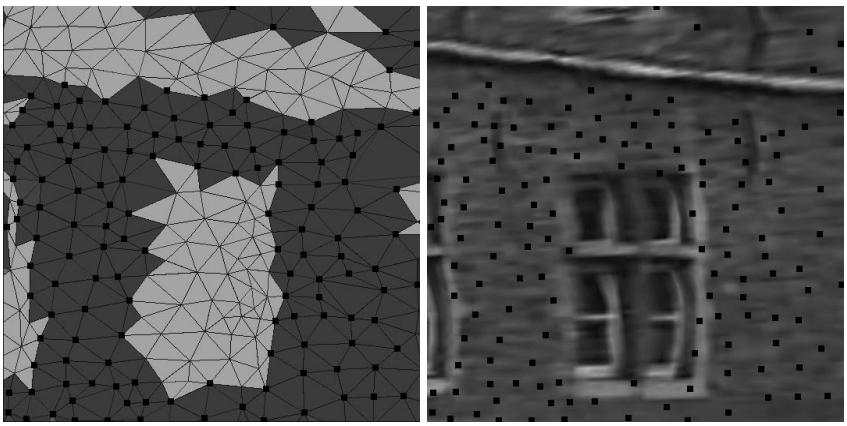

Figure 4.2: On the left is an example of the 3D plane extraction on a wall. The dark triangles represent the extracted plane. The shape of the window in the middle and the pipe on the top are preserved. The mesh vertices with dots are the vertices that lie in the plane and are projected onto the optimised constrained model. On the right is the corresponding texture map with the same vertices.

This clustering starts by creating one cluster for each triangle. The two clusters which have the closest distance to each other are then merged repeatedly. The Euclidean distance between the triangle centroids $C_{1,2} = (c_x, c_y, c_z)'$ is used as the distance measurement:

$$|C_1 - C_2| \tag{4.7}$$

The distance between two clusters is defined as the minimal Euclidean distance of any two triangles belonging to different clusters (nearest neighbour method). Our implementation of the clustering technique and its properties are described in appendix B.

The result of this is that outlying triangles and distinct planes will end up in different clusters. Outliers are removed from the planes and large disconnected planes are split into several smaller planes.

4.2.4 Data mapping

The plane detection algorithm works in the domain of the triangles. However, in the end the vertices of the model are projected onto the optimised constrained model. So, the set of extracted plane triangles is converted into a set of plane vertices. Further work like model optimisation (calculating the fit of the model) is then carried out on the vertices.

Often triangles cut through edges at the border of planes. Such a triangle is likely to be detected as a plane triangle, although not all of its vertices belong to the actual plane, since the triangle cuts through the border. Using every vertex of every triangle on the plane typically makes the plane too large, destroying the plane border. Therefore we use a simple heuristic to map the triangles to the vertices. One can test the vertices at the plane border specifically if they belong to the plane. We use a different approach which removes small holes in the planes and straightens the border of the planes. The neighbouring triangles of each vertex are analysed. A vertex is determined to belong to the plane if the area (triangle sizes) of the surrounding plane triangles that lie in the plane is greater than the area of the surrounding non-plane triangles. This method smooths the plane border. Vertices mainly surrounded by plane triangles become plane vertices and vertices mainly surrounded by non-plane triangles do not become plane vertices. Figure 4.2 shows an example where the mesh vertices with dots have been determined to lie in the plane (dark gray triangles).

4.3 Lines

The other architectural structures that are very common are linear elements. In order to extract them from our data we first create a subset of edges which are likely to lie at jump edges (also known as depth discontinuities) or roof edges (also known as crease edges or fold edges). Line hypotheses are then created on the edges and the ones with the highest rating are selected.

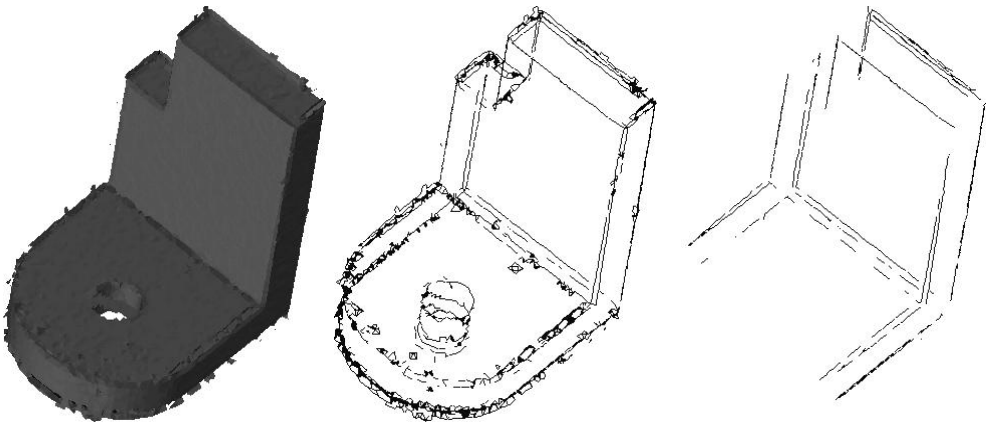

Figure 4.3: This illustrates the line extraction method applied to the 3D model of an industrial part. On the left is the original solid model. It is represented by a noisy triangular mesh with 11675 vertices, 34288 edges and 22556 triangles. In the middle are the the extracted initial edges with 2622 vertices and 3275 edges. On the right are the extracted lines with 843 vertices and 766 edges.

4.3.1 Edge extraction

The edges are computed from the triangular mesh. We start with a list of all mesh edges (*i.e.* edges linking vertices). First, jump edges are extracted. Edges belonging to a single triangle are marked as jump edges. The next step is to extract roof edges. Roof edges separate two non-coplanar triangles. All edges lying in previously-extracted planes are removed from the list. One can further restrict the set of initial edges by only allowing edges which lie at borders of the set of previously extracted planes. Then, for each remaining edge, the two triangles beside the edge are considered and the angle between the surface normals of the two triangles is computed. If the angle is above a certain threshold (we use a relatively conservative threshold of 30 degrees here) then the edge is marked as a roof edge. If not, the the edge is removed from the list. We end up with a initial edge set consisting of jump edges and roof edges (see figure 4.3).

4.3.2 Line hypothesis

The algorithm then creates random lines from the edge set. The minimum sample size for a line are two vertices $V_n = (v_{n,x}, v_{n,y}, v_{n,z})'$ from V. Analogously to the plane extraction, half of the hypotheses are constructed by selecting pairs of random vertices. The other half of the hypotheses are constructing by selecting the two vertices of a single random edge $E_1 = (v_1, v_2)$ from E.

For each line hypothesis, the line parameters are calculated. The first vertex is the line starting point $S = (s_x, s_y, s_z)'$.

$$S = V_1 \tag{4.8}$$

The orientation $O = (o_x, o_y, o_z)'$ of the line is calculated by subtracting the second vertex from the first one and normalizing:

$$O = \frac{V_1 - V_2}{||V_1 - V_2||} \tag{4.9}$$

To make the representation more compact, the line orientation is converted from the vector with x, y and z values into azimuth and elevation angles with respect to the reference vector $(1,0,0)'$. Each line is then represented by the starting point (three floating point numbers) and the orientation (two floating point numbers).

4.3.3 Line evaluation

To find the best line hypothesis, all hypotheses are evaluated. An edge $E(v_1, v_2)$ from the edge set belongs to the line if both vertices of the edge are below a certain distance away from the line. Analogously to the plane extraction, we need to set the distance threshold to a value depending on how much we want to straighten lines or to preserve details in the model. Since all models are normalised, the distance threshold is easily comparable between models.

The distance of a vertex to the hypothetical line is calculated with the starting point $S = (s_x, s_y, s_z)'$ and the unit orientation $O = (o_x, o_y, o_z)'$ of the line. All triangle edges with their vertices (v_1, v_2) that satisfy the following inequality belong to the hypothetical edge (see figure 4.4).

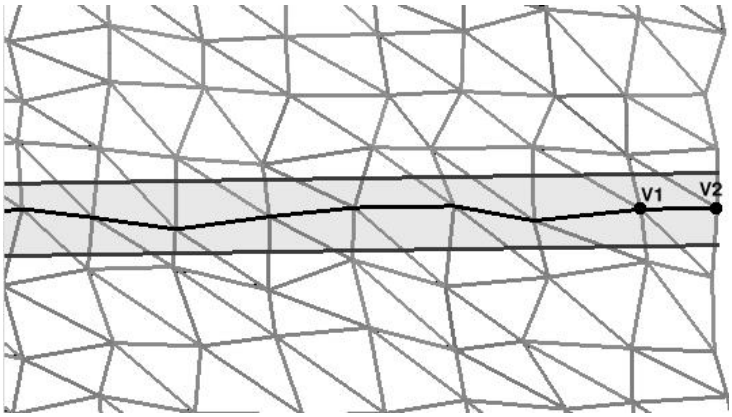

Figure 4.4: A fold edge goes horizontally through this mesh. A hypothetical line is created with the vertices V_1 and V_2. All edges that lie in the 3D corridor (grey area), the width of which is determined by the distance threshold, belong to the hypothetical line (middle dark line).

$$||(V_i - S) - O((V_i - S) \cdot O)|| < tolerance \tag{4.10}$$

The length of a hypothetical line is calculated by adding up the lengths of the matched triangle edges. The hypothesis that gets the most support (*i.e.* creates the longest line) is selected.

Man-made structures such as cars, windows or furniture typically include several hundred linear elements. However, we are only interested in linear elements from architecture. They are typically rather long relative to linear elements in the natural environment or to linear elements in furniture. Therefore, we start with extracting the longest linear element and repeat the process until the length of the resulting line falls under a given threshold. The threshold depends on the amount and size of architecture in the scene. As an example, Figure 4.3 shows first the edge set that is created and then the 3D lines that are extracted from an industrial part rich in linear structure.

4.4 Evaluation

A good feature extraction is essential for the later parts of the constrained reconstruction process such as scene interpretation and model optimisation. Our experiments with sample models address the questions: How does extraction performance change when we change from a 3D point-based plane extraction to a higher 3D object- (*i.e.* triangle-/edge-) based extraction? How sensitive is the extraction to the distance parameter? At which point does the extraction fail?

We evaluate the differences between point-based and triangle-based plane extraction with a model of a single room in an indoor environment. The model we are using here consists of several large planes on walls and furniture (see figure 4.5). It was reconstructed by the European Commission Joint Research Centre (JRC) [SNW+99]. Both extraction techniques create random plane hypotheses. The triangle-based extraction evaluates the hypotheses by summing up the area of all triangles that support them. The extracted planes are represented by sets of triangles. On the other hand, the point-based extraction counts the points that support the hypothetical planes. The extraction results in one set of points per extracted plane. The triangle-based extraction finds the 7 largest planes that are the walls and the ground floor. In contrast, the three largest extracted planes by the point-based method are the ground plane, a plane close to the back wall and a plane going across the furniture in the room. The plane close to the back wall mostly includes the windows and some (but not all) points from the actual wall. The point-based extraction fails for this kind of scene because points on the planes (*e.g.* walls) are very sparse and those on the detailed structure (*e.g.* furniture) are very dense. In such cases, a count of points is not sufficient to extract planes. The triangle-based extraction overcomes this problem by taking the plane area (triangle sizes) into account.

The second evaluation was carried out to show the effect the distance parameter has on the plane and line extraction. With which parameter value can we obtain a good feature extraction? We use two synthetic scenes each consisting of a mesh of three connected planes (1323 vertices & 2400 triangles). The scene for the plane extraction evaluation is created in a way that makes it difficult for the extraction to obtain a perfect segmentation. The three connected planes have only a slightly different orientations

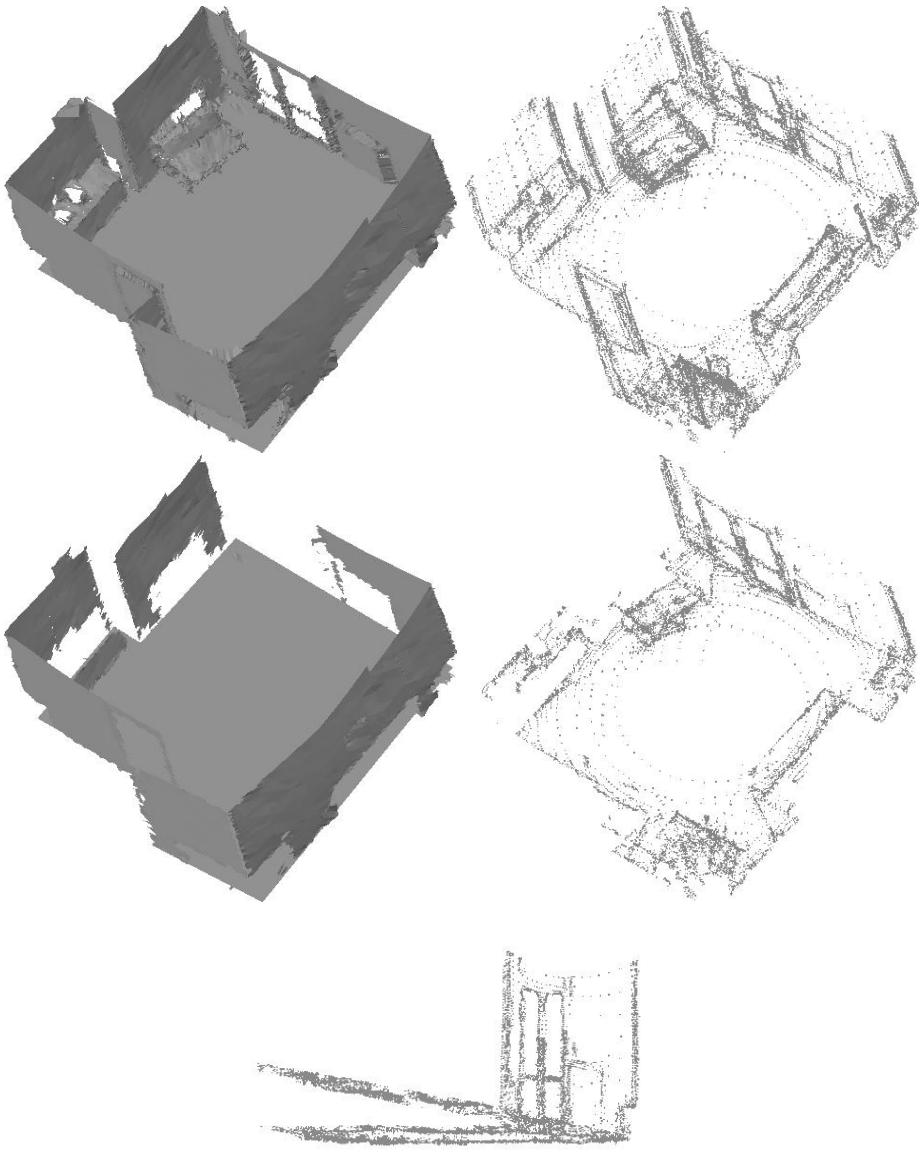

Figure 4.5: The top images show an indoor scene (left: solid, right: points) reconstructed by the European Commission Joint Research Centre (JRC). The 7 planes extracted by the triangle-based method are displayed in the middle on the left. The middle right shows the points belonging to the 3 extracted planes by the point-based method. The bottom shows the result of the point-based method from the side.

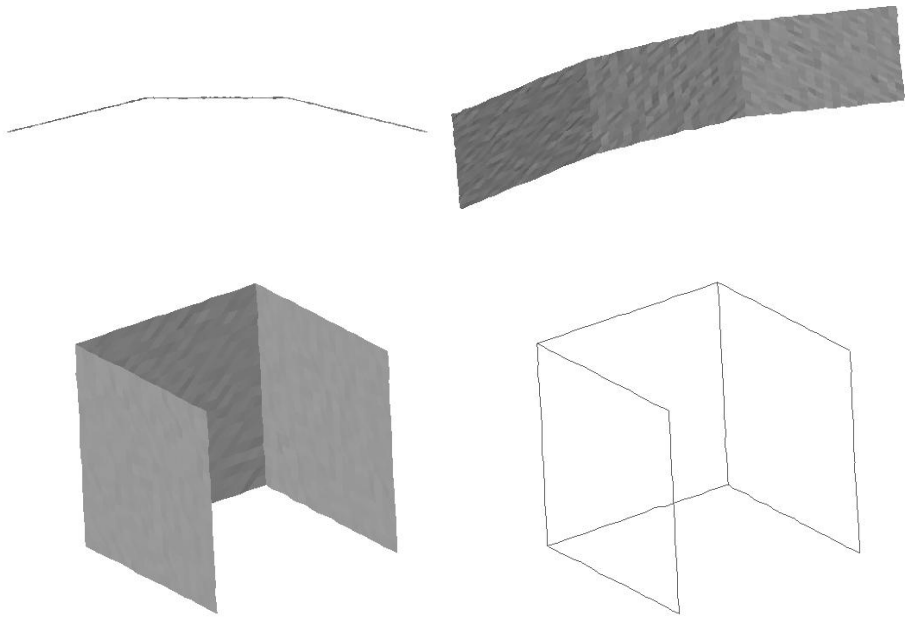

Figure 4.6: Two synthetic meshes are used. On the top is the mesh for the plane extraction evaluation (left: top view, right: front view). On the bottom is the mesh for the line extraction evaluation (left: solid view, right: initial edges).

of 15 degrees (see figure 4.6 top). The planes in the model for the line extraction evaluation are perfectly orthogonal to each other. Two planes are parallel (see figure 4.6 bottom). This makes it easier to pick up the edges of the initial edge set (see section 4.3.1). A small amount of Gaussian-distributed 3D noise with a variance of 0.004 is added to the vertices of the normalised mesh models (see Figure 4.6). Planes and lines are then extracted with different distance thresholds from the model and the results are compared with the known ground truth. The ground truth is a hand extraction, which specifies the triangles for each plane and the edges for each line in the 3D model.

Figure 4.7 shows graphs of the evaluation of the plane and line extraction. The left graph shows the planes recovered by the algorithm as a percentage of the total plane area as the distance threshold is varied. Analogously, the right graph shows the lines recovered as a percentage of the total line length. The small jitters in the curves are a result of the noise and the random nature of RANSAC. With low thresholds, no fea-

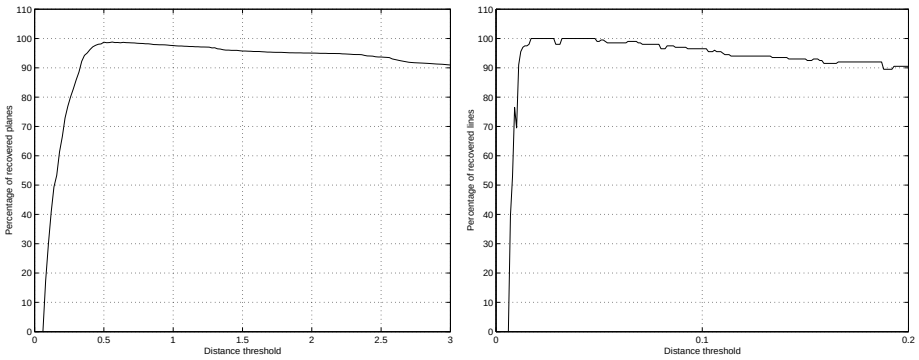

Figure 4.7: The graphs display the percentage of recovered features versus gradually increasing distance thresholds for the plane (left) and line (right) extraction.

tures or only parts of the features are detected. As the threshold increases more and more triangles and edges are extracted as part of planes and lines. After reaching a good extraction (correct extraction is close to 100 percent), the extraction performance graduately decreases in small steps, because parts of planes and lines are extracted as belonging to the incorrect planes and lines. No perfect extraction (only 99 percent) was obtained for the plane extraction, because the triangles close to the plane intersections are not sufficiently far away from the recovered plane. The similar orientation of the planes affects the extraction at the intersection. This results in a small mis-segmentation (*i.e.* single mis-extracted triangles at the intersections) even if the best possible distance threshold is selected. This experiment shows us that feature extraction is fairly stable after reaching the optimal threshold and before the mis-extraction becomes increasingly evident. We can select a distance threshold from a range of stable values and obtain a good segmentation. However, perfect extraction at feature intersections can be difficult depending on how well-distinguished the features are and how much noise is on them.

4.5 Discussion

This chapter discussed how to extract basic primitives such as planes and lines from a triangular mesh, which is necessary in order to apply constraints on them later.

RANSAC techniques have been used to find these primitives.

Classically, RANSAC techniques count the number of points supporting the hypothetical model. Here, we can exploit the properties of the triangular mesh. Triangle sizes and edge lengths give a much better quality estimation of the hypothetical plane and line than a 3D point cloud could support alone. Many triangular meshes use relatively few big triangles (*i.e.* few points) to model planar surfaces like walls and many small triangles (*i.e.* many points) to model detailed structure like furniture. Relying only on the 3D points to extract planes and lines in such a model would ultimately result in omitting the walls and detecting the furniture, since the hypothesis evaluation can only count the number of points, which are dense at the furniture, to judge how good the hypothesis is.

Post-processing is necessary, especially for planes. In more complex real world scenes, a plane cutting through the 3D model includes some elements that are not part of the plane we desire to extract. Elements far away from the plane belonging to other structures are detected using single linkage clustering. In contrast to planes, extracted lines are not so sensitive to the inclusion of erroneous edges. Since a line is a 1D structure and both vertices of a potential edge must lie on it, artifacts are less likely to occur.

Typically, after finding a model with RANSAC, the estimated model parameters are reestimated by fitting the data to the model. The first estimates can thus be improved. In our case however, recomputing the model parameters is not necessary, since the parameters are improved in the optimisation step later on parallel to the application of the geometrical constraints.

Future work on feature extraction could take the mesh texture into account. Most meshes include colour information in the form of a texture map. Texture might not be available in some cases, especially where coplanarity constraints on the 3D data are used to make the texture mapping easier to estimate. However, with colour information the line extraction could be improved as described in [DSGV01]. They extract lines in the texture, which allows the line extraction in the 3D data to be done more reliably. Similarly, one could improve the planar segmentation as well.

Chapter 5

Scene interpretation

This chapter discusses the automatic interpretation of the architectural scene represented by planes and lines. We use planes and lines extracted as in the previous chapter. The interpretation stage enables us to assign appropriate geometric constraints to relationships that are nearly regular such as nearly-parallel walls or linear elements. The constraints between features are then used to enforce parallelism or orthogonality later in the optimisation process in chapter 6.

This chapter starts with reviewing briefly scene interpretation methods. After this, we discuss how to match the planes against a model of a generic house. The next section shows how to analyse the extracted lines. The principal directions are obtained from the set of lines. Finally, the chapter concludes with a discussion of the methods and their limitations.

5.1 Previous work

Few papers have dealt with the automatic extraction of constraints - most leave it to the user to specify them. The majority of approaches expect that the user will define constraints within an interactive user interface [FRL+98, SM99, RC00], in a constraint definition file [WFAR99b, RFWA00] or even in the source code. The constraint definition file in [WFAR99b, RFWA00] is a list of statements in which the user declares the surfaces such as planes, cylinders or spheres, their parameters and the constraints with

their associated values and tolerances. These approaches require heavy human interaction, because constraints and their attributes must be selected manually for every new scene. On the positive side, they allow fairly complex constraints to be used and object recognition and classification is fairly accurate, since a human being provides it.

Obtaining the feature labels and the relationships between them automatically is challenging. Architectural features can be identified by using statistical methods like Markov Decision Process [Dra96], Markov random fields (MRF) [Bru00] or Bayesian models [MJHR99, TDC00, DTRC01]. The statistical models used by these statistical techniques need to be known a priori; or automatically learnt.

Beside statistical methods, features can also be identified by matching them against a knowledge base. The classical knowledge base in image analysis is VISIONS [HR78]. It uses a hierarchical model (pyramid model) where each layer represents image properties at successively coarser resolution and relates only with properties in layers immediately above or below or within the layer. The highest layer for example includes complex objects. These objects are broken down in the lower layers to volumes, surfaces, regions and single points.

Liedtke used such a knowledge base in the form of a semantic net for interpretation of architectural scenes [LGG95]. His semantic net is organised in three hierarchical layers. The lowest layer includes image points (the sensor data). The middle layer includes 3D data like polygons and vertices. The highest layer represents high level 3D objects like “walls”. The interpretation is hypothesis driven. Hypotheses are verified or disproved by matching the hypothetical 3D objects against the 2D image.

One does not have to identify high level features like “wall” or “window”. Instead, the model can be directly searched for shape regularities. Especially for reverse engineered industrial models, it is better to search for regularities [LMMM01] and also for symmetries [MLMM01]. A meaningful labelling may be impossible to obtain since the design of industrial objects differs widely and the general concept of object parts such as “wall” or “roof” are not present. For example, Langbein shows how to extract shape regularities from reverse engineered models [LMMM01]. He assumes that the regularities are sufficiently distinct from the noise in the models. He uses hierarchical clustering to identify groups of features with similar parameters. Parallel axis direc-

tions, like normals of planes, are identified for example. Thus, regularities between features are revealed. However, for extracting regualities from architectural models it is better to label first the objects and then search for regularities [DTRC01]. The identification of regularities and thus the selection of suitable geometrical constraints is driven by the interpretation of the scene.

5.2 Plane labelling

We describe a house by its generic parts and the relations between them. These relations are modelled as constraints between model parameters of different house parts. We obtain the identification of the house parts (*i.e.* labels) and the relationships between them by matching the planes against a semantic net of a generic house. By plane identification we mean the selection of a class of architectural feature (wall, floor, ceiling, roof, etc.) to which a plane belongs. The semantic interpretation of generic house parts is then used to select appropriate geometric constraints to enforce for example parallel or perpendicular walls.

The plane labelling is formalised as a constraint satisfaction problem (CSP). The CSP is to determine the n-tuples of value assignments that are compatible with all constraints (*i.e.* model relationships). CSP is a fundamental problem in Artificial Intelligence, with applications ranging from scene labelling to scheduling and knowledge representation. In computer vision the CSP approach dates back to Waltz [Wal72] and Mackworth [Mac77]. Waltz showed how to obtain a consistent three-dimensional interpretation of line drawings by exploiting model relationships. He demonstrated how backtracking tree search can be improved. The search problem can be solved without encountering any dead ends, when constraint propagation in the form of arc consistency is applied.

Our work matches the planes against the semantic net by using a classical backtracking tree search. The semantic net concentrates on the definition of the 3D objects and their relations. It is similar to Liedtke's highest layer which represents the high level 3D objects. Liedtke used the semantic net to interpret intensity images of architectural scenes. The novelty in our approach is that we interpret by checking the

relationships between the 3D objects. The geometric reasoning in 3D space leads to the selection of the correct geometric constraints. For example, a plane parallel to the roof does not occur in our model of a generic house. Thus, no constraint is used to enforce such a relationship.

5.2.1 Model of a generic house

Knowledge about structures can be represented efficiently by semantic nets. Semantic nets consist of nodes (also called labels) and arcs between them. Here, the knowledge of a generic house model is represented explicitly in a semantic net (see figure 5.1). The model entities (walls, roof and floor) are represented as nodes in the net. The nodes are connected via different types of relationships (arcs). The model-entities (labels) and the relationships among the entities represent the knowledge of a typical architectural scene of a house. The domain of possible nodes is L = {*Side Wall, End Wall, Base Plane, Ceiling/Floor, Roof*}.

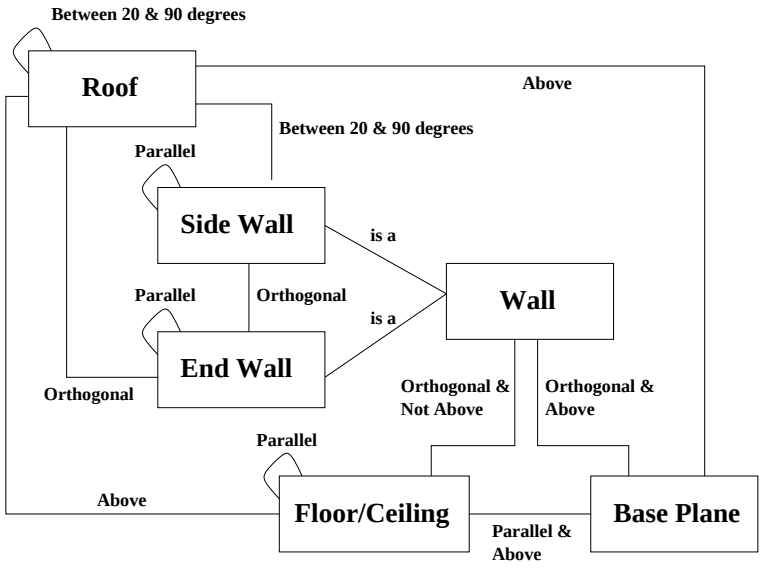

Figure 5.1: The model of the architectural scene is represented by a semantic net. Nodes represent the model entities and are linked by architecturally-meaningful relationships.

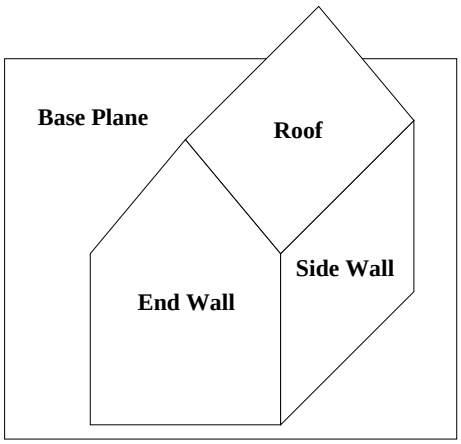

Figure 5.2: The house model consists of planar differently oriented surfaces that are on top of a base plane.

Each label is connected by given arcs to other labels. The arcs express particular geometric relationships between the labels. This set of relationships is used to constrain the matching. The *Base Plane* is below all other planes. It represents the ground on which the house stands. The plane is parallel to all planes of type *Floor/Ceiling*. The *Base Plane* and *Floor/Ceiling* are orthogonal to all walls. Two kinds of wall are possible. The *End Wall* is at the beginning and end of the house and the roof. The *Side Wall* goes beside the roof (see figure 5.2). The two wall kinds are orthogonal to each other. Finally, the roof consists of one plane or two planes that are between 20 and 90 degrees to each other. The roof is modelled as a typical peaked roof. All other planes are below the *Roof*. The *Roof* is orthogonal to the *End Wall* and between 20 and 90 degrees to the *Side Wall*. Figure 5.2 illustrates the different orientations of the planes.

The semantic net models a reasonable subset of all houses. It includes the interior and exterior structure of houses. The model can include an arbitrary number of walls which can be placed parallel or orthogonal to each other. Walls can form several rooms. They can be on the same level or on different levels (then separated by a *Floor/Ceiling*).

5.2.2 Labelling

The input data from the scene to be interpreted consists of a set of planar surfaces. A backtracking tree search is used to find the best match of the data to the house model.

The algorithm takes as input a set of plane features F, a set of possible model labels L and a set of binary model relationships R which limits the possible labelling. The tree search starts with the first feature from F and assigns all labels from L. A second feature is fetched from F and all labels are assigned. At this level a label might be ruled out because it violates the observed scene relationships. This leads to a dead end in the tree search. So, we go up and continue with the next possible labelling. This process continues until all features have been labelled (see figure 5.3). A consistent labelling then exists if we find an instance of the labelling that satisfies all relationships. That means that each plane is assigned a single valid label that is also arc-consistent with adjacent nodes.

We use explicit geometric information to constrain the matching process. The architectural relationship between two features and their labels are checked (*e.g.* horizontal and vertical walls are almost perpendicular or the base plane is below all other features). Angle relationships between two features are checked with a certain tolerance. This angle tolerance depends on the possible deviation parallel and orthogonal planes are allowed to have in the scene. The “Above” relationship is satisfied if the vertices of one plane are above a second plane defined by surface normal and distance. In the case that the two planes are orthogonal to each other, not all vertices might be above the plane, since a small percentage of the other plane might lie on the same level. Thus, only 99% must lie above.

Some extracted planes from the scene might not be part of the house. They might belong to cars, furniture or other houses. Such planes typically have no satisfactory relationship with any plane from the house. Therefore, no valid label can be assigned to them. For these planes the *No Architectural Feature* label is used. Thus, possible inconsistencies can be resolved. The final labelling is obtained by finding the solution that maximises the number of valid architectural labels. However, it may happen that several competing labellings are found. This situation appears if we do not have enough data. For example, a scene consisting of only two orthogonal planes can be la-

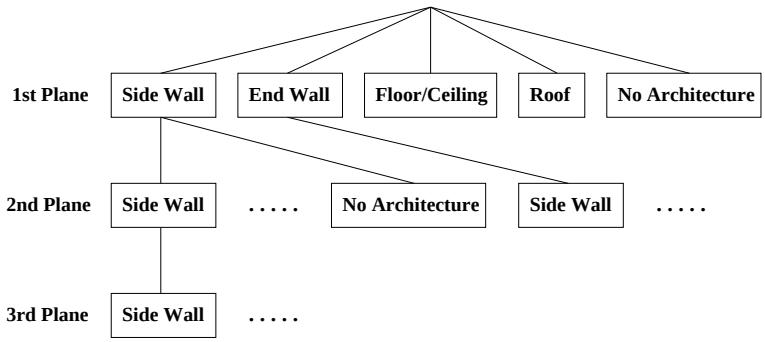

Figure 5.3: The labelling of the features is obtained by a backtracking tree search. The first plane is labelled with all possible labels. The other planes are labelled, but some label combinations are ruled out because they violate the given relationships.

belled in multiple ways such as wall-floor or wall-wall. The relationships between the planes and therewith the constraints remain the same, but the labelling is ambiguous.

5.3 Line grouping

Architectural scenes include much linear structure in the form of parallel and orthogonal lines. In particular, the lines of an architectural scene form a structural frame defined by the three principal directions (defined in section 3.1.2). The new contribution here is to use the principal directions to improve the orientations of the lines in the structural frame in our architectural models. Recently, the principal directions has also been used by Werner [WZ02b] to reconstruct a planar model of architectural scenes from intensity images. In 3D geometry the analogous term to the principal directions is the normalon [PCM99]. An object in 3D space is a normalon if all its surface normals are aligned with one of three mutually-perpendicular axes.

We obtain groups of parallel lines by hierarchical clustering them. Feature clustering has been used before for example for detecting redundant features [BD01] and identifying regularities [LMMM01]. The regularity we are looking for, the three principal directions, are likely to include many of the lines from the scene. Therefore, we cluster first and then search for the three clusters which reflect the principal three di-

rections of the architectural structure in the extracted clusters of parallel lines. Three orthogonal constraints are created to enforce the directions.

5.3.1 Clustering

Each line has an orientation O and a starting point S. For obtaining groups of parallel lines we look for lines with similar orientations. Lines with almost-equal orientation are grouped together by complete linkage clustering [JD88], as follows. Initially, each line represents its own cluster. The two most similar clusters are joined at each step. The distance between clusters is determined by the greatest angle between the orientations of any two lines in different clusters (opposite to the nearest neighbour method). This method leads to very compact clusters. We stop the cluster process when the next two clusters selected to get joined are too dissimilar (*i.e.* the angle between these two clusters is greater than 2.5 degrees). We use the angle between the normalised orientations O_1 and O_2 of the lines as the similarity measurement:

$$acos(|O_1 \cdot O_2|) \tag{5.1}$$

The clustering reveals groups of lines that are considerably closer to each other than to the other clusters. Cluster techniques are described in more detail in appendix B. The final orientation of each cluster is the average orientation of all its lines weighted by their lengths.

5.3.2 Principal directions

The majority of architectural structures consist of linear elements that form a three-dimensional structural frame [Chi96]. The frame defines the three principal directions of a building. Figure 5.4 shows as an example the extraction of the lines and in particular the lines in the principal directions in a factory scene.

We use the clusters found previously to find the three directions of the architectural frame. We calculate the angle error between all triple combinations of clusters O_1, O_2 and O_3. We find the three directions from the set of clusters by minimising the angle error:

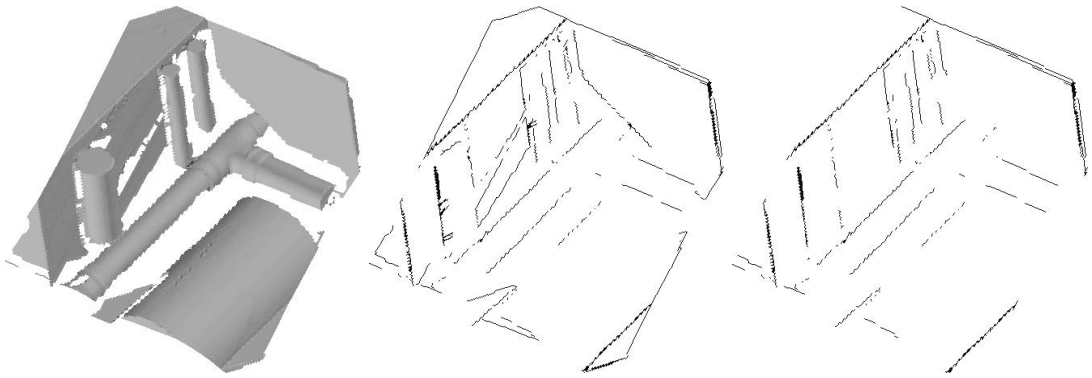

Figure 5.4: This is an example of the three principal directions in a reconstruction of a miniature factory scene (left). The scene includes two walls, several pipes along the walls and a big cylindrical object in the foreground. The extracted lines are shown in the middle. The subset of lines in the three principal directions is shown on the right.

$$\sum_{i=1}^{2}\sum_{j=i+1}^{3}(|\frac{\pi}{2}-\arccos(\frac{|O_i\cdot O_j|}{\parallel O_i \parallel \ \parallel O_j \parallel})|) \tag{5.2}$$

The clusters from the combination that most closely resembles the principal directions are selected. Orthogonality constraints are created between the orientations and these are used in the optimisation process. However, the principal directions are not necessarily present in every architectural scene. Architecture can be designed intentionally without following the principal directions. Therefore, we need to check the angle error of extracted clusters to determine whether the principal directions are present in the scene.

5.4 Discussion

This chapter has discussed the interpretation of the scene. The planes are labelled and the relationships between them are obtained. Lines are clustered to find those parallel to each other. Additionally, the three principal directions are extracted from them.

A semantic net of a generic house is used for the plane labelling. The semantic net can be easily extended with features like windows and doors. These features can

be modelled as parallel and close to the actual walls. However, the previous plane detection concentrates on finding big planes. So, modelling windows and doors is not possible at this stage.

The backtracking tree search tries all labels on all features. Invalid features are marked as *No Architectural Feature*. However, as mentioned before, the plane extraction concentrates on finding large planes. So it is unlikely to find non architectural planes, which are likely to be small. However, extraction of planes from other adjacent houses is still possible.

Not all architectural scenes can be labelled with this technique. The labelling for a scene which includes (say) only three walls might be ambiguous. Several labellings (like “wall”, “ground plane”, “ceiling”) might be possible for one plane. It is hard to find the general orientation of the scene in this case. What is above/below? However, a roof with its special geometric relationships inside the scene usually gives an important clue about the global orientation. It enforces the labelling of all planes below as walls, ceilings and so on.

The model can be used to interpret interior architectural scenes as well. Such structure could include $\{$*Floor/Ceiling*$\}$, $\{$*Side Wall*$\}$, $\{$*End Wall*$\}$. However, we then encounter the problem that we do not know the global orientation of the scene mentioned before.

The three principal directions are extracted by cluster analyses of the lines. However, we need to check if the principal directions are really present in the scene by checking the angle error of the extracted clusters. Moreover, complex architectural models (*i.e.* models including whole or even several buildings) might include several independent structural architectural frames (*e.g.* one for each building). Each of these frames has its own three principal directions. In our case we extract only one set of principal directions, since our models are not as complex.

Constraints are assigned to near-regularities such as walls and lines which are approximately parallel or orthogonal. The number of constraints on the lines is relatively small compared with those on the planes. Constraints between individual line orientations are not created. In a scene with many hundred lines, this would lead to many hundreds of constraints and the scene would be greatly overconstrained. Overconstrained

models are computationally more expensive to optimise. Moreover, the weight of the constraints relative to the fit during optimisation would be much higher resulting in a bad fit of the data to the model. Therefore, we define that all parallel lines have the same orientation. Then, only three orthogonality constraints to enforce the principal directions are assigned.

Future work on labeling planes could take texture into account similar to the future direction for feature extraction. 3D features and the 2D texture has been used to improve terrain classification from aerial images [WSS+97]. In a reconstructed architectural environment one could exploit that for example house roofs have a certain texture.

Chapter 6

Model optimisation

The previous chapters were concerned with the identification of features and regualities between those features. This chapter presents the final step, that of applying architectural constraints to the model. The original triangulated model is transformed to the new constrained model, which enforces the regularities by applying constraints.

The computation of the new constrained model is an optimisation problem. Most real world optimisation problems are complex and loosely-defined and therefore need some simplifying assumptions in order to become solvable. Often a reasonable solution is sufficient but in some other cases the absolute global optimum must be found. Some optimisation techniques become ill-conditioned and fail when applied to non-linear problems. Here, we use two different robust optimisation algorithms, a genetic algorithm (GA) [CFD02b] and the Downhill Simplex algorithm [CFD02a], to solve the optimisation problem and thus to obtain the constrained model. Our contribution is to use the Downhill Simplex for constrained optimisation of geometrical 3D models and to perform a comparison with the well-known GA approach [RFWA00, RF02].

The chapter starts with a description of the optimisation process (Section 6.1). After finding the model parameters for the constrained model the model vertices are projected onto the now optimised (with regard to orientation and position) features (Section 6.2). The result is a model with well-aligned features (*e.g.* parallel walls) and fewer irregularities (*e.g.* edges on walls). At the end of the chapter, we evaluate the two optimisation algorithms used at the optimisation process (Section 6.3).

6.1 Optimisation process

The constraints[1] found previously are enforced by constrained optimisation. The constrained optimisation aims to find the optimal model parameter values which are close to the original data and at the same time satisfy the architectural constraints. An evaluation function is used to measure how good the support of the parameters to the model and the constraints is. The optimisation involves finding the optimum (in our case the minimum) of the evaluation function with respect to the model parameters.

The optimum of an evaluation function can either be global, that is the highest or lowest value over the whole region of interest; or local, that is the highest or lowest value over some small neighbourhood. More complex evaluation functions typically have many local optima, whereas we are usually most interested in finding the global optimum. This can be very difficult. The optimisation algorithms cannot distinguish whether we are at the global optimum or at a local optimum, making it difficult to decide when to stop the search and when to carry it on to find a better optimum. In particular, local optimisation techniques head directly towards the "next" local optima, so ignoring the global search space with potentially other better optima.

Optimisation problems have been studied extensively, reaching back as far as Sir Isaac Newton (1643). Many algorithms have been developed to search spaces for an optimum solution. The simplest optimisation problems work in one dimensional space, optimising only a single variable. Multi-dimensional problems are much harder. For example, in a two-dimensional minimisation we need to find the instances for two variables that specify the point at the bottom of the lowest valley in the landscape of the two dimensional search space (see figure 6.1 for an example landscape). The many algorithms which are possible differ very much in respect to the sensitivity to local optima and computational speed. Classical optimisation methods for finding local optima are the simple hill climber, Powell's method, the gradient-based hill climber using the first derivatives (also called Quasi-Newton methods) and the Levenberg-Marquardt method using the second derivatives. The simple hill climber moves slowly in the optimal direction. This requires all directions to be tested. Derivative-based methods are computationally more efficient. They jump close to the estimated optimal posi-

[1] parallelism and orthogonality constraints between lines and planes

tion. However, derivative-based methods greatly constrain the evaluation function. The derivative of the function must exist. Furthermore, all these methods can easily get stuck in a local optimum and are sensitive to noise.

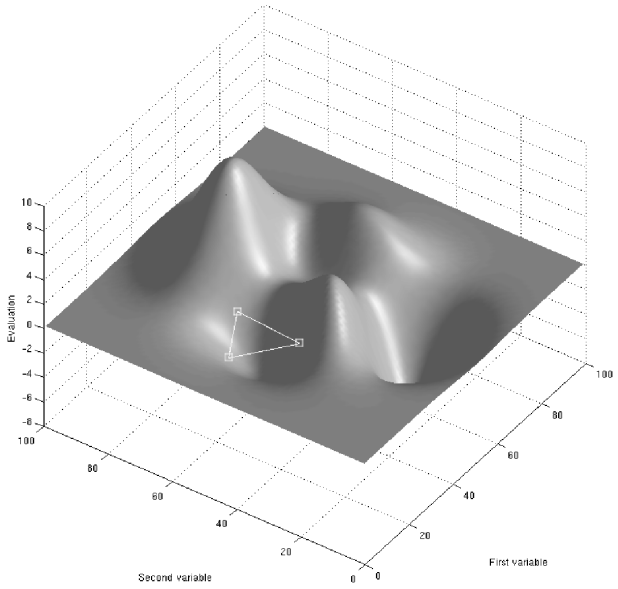

Figure 6.1: This image shows a landscape of a two dimensional search space. The vertical axis corresponds with the evaluation of the points in the search space. A simplex in form of a triangle (3 points and each point with 2 variables) crawls into a depression in the front.

Global optimisation techniques are useful when the search space is likely to have many optima, making it hard to locate the true global optimum. The simulated annealing and genetic algorithm (GA) approaches are specially designed to search the global search space and are thus more likely to find the true global optimum. The Downhill Simplex algorithm [NM65] is not strictly 'global' (in the sense that it searches the full parameter space), but it is able to crawl out of some local optima to find better optima.

Some of these optimisation techniques have been used for optimising reconstructed 3D models. Gradient-based hill climbers have been used in [WG96, LMM02]. More complex techniques such as the Levenberg-Marquardt method [WFAR99b] and GA [RFWA00, RF02] have been utilised as well. We chose two quasi-global optimisation techniques, a GA [CFD02b] and the Downhill Simplex method [CFD02a], because

they require only function evaluations, not derivatives, thus giving us freedom in designing the evaluation function. We may, for instance, incorporate non-differentiable operators such as the absolute value into our evaluation function formula. Unlike derivative-based optimisation methods, both methods are very robust. Like any technique which uses only function evaluations, these two techniques are not very efficient in terms of computational performance. However, we will show that the Simplex method leads to a quick solution if we already know a set of parameter values for a solution close to the optimum - as is the case in our situation.

6.1.1 Model representation

In the feature extraction process (Chapter 4), we extracted lines and planes from our data, allowing us to represent our model by a vector $\vec{p}$ of independent model parameters. This vector is a concatenation of all n parameters for the individual lines and planes.

The geometric parameters for each plane include the surface normal and the distance from the origin (see table 6.1). A line is represented by a starting point and an orientation. The distance is represented by one value (one float), the starting point by the x, y and z values (three floats) and the angles (*i.e.* surface normal and orientation) by azimuth and elevation angles (two floats) with respect to a reference vector. Azimuth and elevation angles are traditional, but suffer from a singularity at the poles. This means that differences of the azimuth angle have different effects depending on the elevation angle. By keeping the number of parameters for each individual feature small, the size of the parameter vector is kept small and so gives better computational performance at finding the optimal parameter vector.

6.1.2 Evaluation function

The optimisation algorithm we use needs to evaluate points in the parameter search space. This is similar to chess programmes which evaluate the quality of chess positions by for example considering the balance of material on the board. Every possible position on the chess board corresponds with one point in the search space. The chess

Model parameter	Parameter size
Plane surface normal	2 floating point numbers
Plane distance to origin	1 floating point number
Line starting point	3 floating point numbers
Line orientation	2 floating point numbers

Table 6.1: The geometric model parameters and their size

program consequently plays the manoeuvre which leads to the board position with the best evaluation.

Our evaluation is performed by calling the evaluation function at a certain point in search space, which is a model instance represented by a certain parameter vector. The function measures the support of the data for the model instance, that is, the evaluation function encodes how well the data fits to the model (squared residuals of the vertices) and how well the constraints are satisfied (constraint penalty functions). We obtain a good result if the evaluation function result is low. Thus, our optimal solution is a minimum.

The support of the data for the model is measured using the sum of the squared residuals [BKV+02]. The squared residual is the squared geometric distance from the vertices to their feature (plane or line). We have a set of parameterised features $F = \{F_i(\vec{p})\}$. Each feature i has a set of vertices $\{V_{i,j}\}$. The residuals for each feature F_i are summed up and normalised with the feature's number of vertices N_i. Thus, feature size does not affect results. $SR(\vec{p})$ is the sum of the residuals from all model features.

$$SR(\vec{p}) = \sum_i \frac{1}{N_i} \sum_j dist(F_i(\vec{p}), V_{i,j})^2 \tag{6.1}$$

We set *a priori* limitations on the allowable values of the independent model parameters by using constraints. Every constraint is represented by a constraint function $c()$. The values of these functions correspond to the degree that the constraints are satisfied. The constraint function can be seen as a penalty function. Unsatisfied constraints penalise the evaluation value (*i.e.* the value becomes higher). This way the optimisation

is gently suggested to move in a certain way to satisfy the constraints. The constraint functions for enforcing orthogonality and parallelism of the orientations O_i and O_j are:

$$c_{orthogonal}(\vec{p}) = |\frac{\pi}{2} - \arccos(\frac{|O_i \cdot O_j|}{\| O_i \| \quad \| O_j \|})| \tag{6.2}$$

$$c_{parallel}(\vec{p}) = |\arccos(\frac{|O_i \cdot O_j|}{\| O_i \| \quad \| O_j \|})| \tag{6.3}$$

We might not want to enforce perfect alignments, though (see section 3.2.3). Therefore, we give our constraint function a certain allowance in form of an angle tolerance. A constraint jumps into action if the angle error is larger then tolerance t:

$$c_{(}\vec{p}) = \begin{cases} c_{(}\vec{p}) - t & if c_{(}\vec{p}) > t \\ 0 & otherwise \end{cases} \tag{6.4}$$

The constraint functions are summed up to give the global constraint error, $C(\vec{p})$:

$$C(\vec{p}) = \sum_i c^{(i)}(\vec{p}) \tag{6.5}$$

The evaluation function to be minimised is then the sum of the squared residuals and the constraint error. λ is a weight factor which scales the constraints to the residuals. We have here a trade-off between fitting and constraining tightly. λ can be made arbitrarily large to make the constraint satisfaction more exact. However, keeping λ small allows a better fit to the original data to be found.

$$SR(\vec{p}) + \lambda C(\vec{p}) \tag{6.6}$$

6.1.3 Genetic algorithm

Genetic Algorithms (GA) attempt to minimize functions using an approach analogous to evolution and natural selection. The *chromosome*, in nature the medium holding the information of the whole organism, encodes for the GA one specific point in the search space. It consists of small blocks called *genes*. The parameter vector $\vec{p}$ which concatenates all the parameters for the individual features is our chromosome.

We use the GENOCOP (GEnetic algorithm for Numerical Optimization for COnstrained Problems) algorithm version 5 developed by Michalewicz [Mic96]. It is a GA which uses real-valued genes and includes methods to deal with linear, non-linear, inequality and domain constraints. One gene corresponds to one floating point number.

In general, a GA maintains a *population* of N chromosomes/search points rather than having just a single point. The first generation is scattered randomly in the search space. Each chromosome is evaluated by the evaluation function and the chromosomes are ranked based on the evaluation. Bad chromosomes are discarded and new ones are created by combining existing good chromosomes (*crossover*). Two *parent* chromosomes are each cut at a random location and the opposing sections rejoined to form two *children*. This allows the good chromosomes to survive and "breed". In addition a small amount of mutation is introduced, randomly changing genes in the chromosomes. Optimal solutions are evolved by iteratively producing new generations of chromosomes in which good solutions are combined (bred) and bad ones are discarded.

The assumption is made that the optima can be formed by combining small subsections of the chromosomes, each of which will tend to improve the function evaluation of any full chromosome containing them. Thus these good building blocks will tend to be reproduced and will propagate through the generations. Eventually, all the individual chromosomes of the current generation should end up in the global optimum. The GENOCOP algorithm stops after reaching a preset number of generations.

Additionally, GENOCOP-specific constraints are used to narrow the search space. Domain constraints are applied to angle, position and distance parameters in the chromosomes. None of the position and distance parameters can ever be outside the range [-1,+1] since the 3D model is mapped into a normal sphere at the origin.

6.1.4 Downhill simplex

The Simplex algorithm [NM65] is an elegant and robust method for function minimisation. It works locally, but is able to avoid some local optima. It requires only function evaluations, not derivatives.

This method uses the geometrical properties of a non-degenerate simplex (*i.e.* a

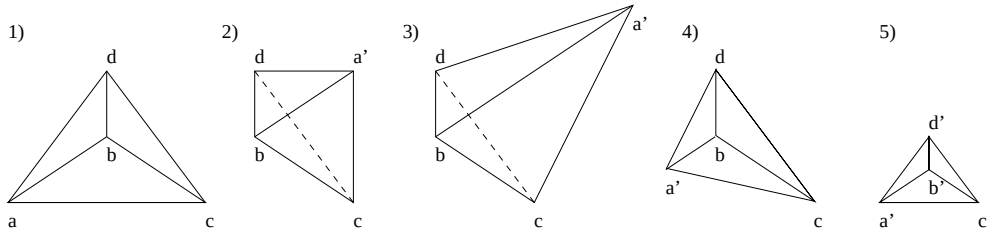

Figure 6.2: This images show a simplex in three dimensions (a tetrahedron). The algorithm will try to move the highest point a of the initial simplex (1). It will either (2) reflect the highest point to a', (3) reflect and expand the highest point to a', (4) contract the highest point to a' or (5) shrink the whole simplex towards the lowest point c.

simplex with a non-zero volume). A simplex in an N dimensional space consists of $N+1$ points and all their interconnecting line segments. In two dimensions, a simplex is a triangle (figure 6.1) and in three dimensions it is a tetrahedron (figure 6.2). In three or more dimensions, a simplex encloses a finite volume. For function minimisation purposes, each geometrical point of the simplex corresponds to a point in the search space. If any point of the simplex is taken as the origin, then the N other points define vector directions that span the N-dimensional vector space. The Downhill Simplex must be started with $N+1$ points, defining an initial simplex. Each point is an N-vector of independent variables.

The algorithm is supposed to make its own way downhill through the complexity of an N-dimensional topography, until it reaches a local optimum. The algorithm starts by creating a random simplex with random points in the search space. If we already know the model parameters for a model close to optimum, we can use this point in search space and place the points of the simplex randomly around it. The simplex then moves iteratively towards the minimum. First, the points of the simplex are ranked based on the result values of the evaluation function. At each step the Downhill Simplex replaces the point with the highest value (highest point) in one of several ways. The algorithm tries first to move the point though the opposite face of the simplex to a lower point (see figure 6.2). This move is called reflection (2). The method expands the simplex in one or another direction to take larger steps (3). If it can not replace the point by a lower point, the method contracts the highest point (4) or if this does not help then all

of its points contract towards the lowest (best) point to shrink the overall volume (5). If the algorithm reaches a valley floor, it contracts all points outside of the valley and tries to move down the valley. When the simplex is trying to “pass through the eye of a needle” it contracts itself around its lowest point and moves through the eye. The algorithm stops after the evaluation values of the search points converge.

The overall effect is for the simplex to crawl around the parameter space, creeping down valleys and shrinking to get to the very bottom of narrow valleys. Figure 6.1 shows an image of a two dimensional simplex crawling into a depression. The use of a set of search points in form of a simplex, that are reflected and contracted, makes the Downhill Simplex very robust and the method of choice for noisy functions.

6.2 Point projection

Point projection is used to transform the original model into the optimised model. Planes are flattened and lines are straightened by projecting their constituent vertices onto the optimised planes and lines. Figure 6.3 shows the projection of line and plane vertices from a simple synthetic scene consisting of two connected planes. We show in Chapter 7 how architectural 3D models containing much architectural structure such as walls, floors and linear elements are optimised. The vertex projection works within the mesh topology, so that when changing a vertex’s position, as a consequence mesh edges and triangles are changed as well. By projecting points, not only are planes flattened and lines straightened, but the orientations of the features are corrected at the same time, since we use the constrained model description for the projection.

6.2.1 Planes

The planes are flattened by a 3D orthogonal projection of the plane vertices onto the nearest point on their plane. We calculate the new coordinates $V_p = (x_p, y_p, z_p)'$ of the vertex with the original vertex $V = (x, y, z)'$, the unit surface normal of the plane $N = (n_x, n_y, n_z)'$ and the distance D of the plane to the origin as:

$$V_p = V - (V \cdot N - D)N \tag{6.7}$$

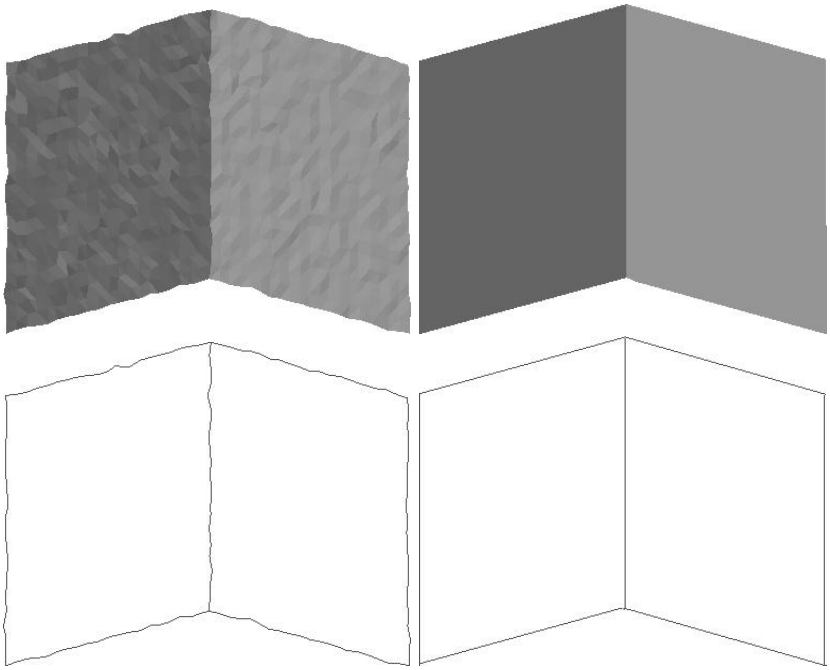

Figure 6.3: The planes of the original mesh model (top left) are flattened (top right) and the extracted lines (bottom left) are straightened (bottom right).

6.2.2 Lines

Analogously, the lines are straightened by a 3D orthogonal projection of the line vertices onto the nearest point of their line. The new position V_p of the vertex is calculated with the original vertex V, the start vertex of the line $S = (s_x, s_y, s_z)'$ and the unit orientation of the line $O = (o_x, o_y, o_z)'$ as:

$$V_p = O((V - S) \cdot O) + S \tag{6.8}$$

6.3 Evaluation

The extent to which different optimisation algorithms are prone to getting stuck in local optima varies. How then do the two algorithms we considered in section 6.1 (GA and the Downhill Simplex algorithm) perform? Do they get stuck in a local optimum far away from the global optimum? Do the two algorithms converge to the

same result? How stable (repeatable) are the results? How does each optimisation algorithm's optimisation performance and computational speed change if the size of the parameter vector increases?

We compare the convergence of the results by executing both algorithms 1000 times with the same data. Every execution performs slightly differently, because of the random starting points and crossover of the GA and the random placement of the simplex around the starting point we obtained from the feature extraction. Additionally, we evaluate the Simplex technique with totally random starting points.

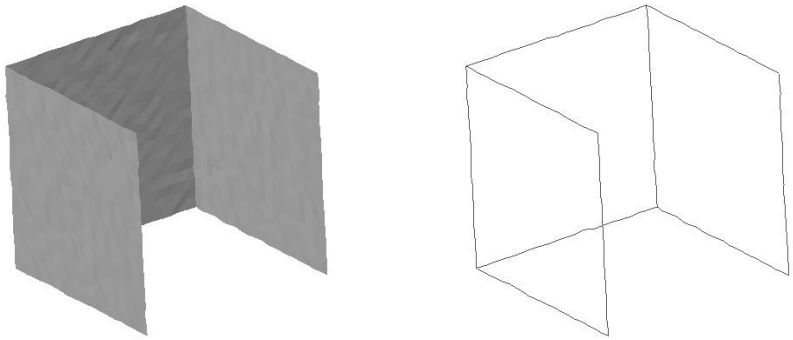

Figure 6.4: The image shows the evaluation scene consisting of three planes at 90 degrees with additional noise added to the vertices on the left. On the right are the 10 extracted lines of the model.

Termination criteria are difficult to define in any multidimensional optimisation technique. The Simplex technique stops if the fractional convergence tolerance of the evaluation function is close to the machine's floating-point precision. The GA stops after a certain number of generations has been reached. After 600 generations are reached we obtain a reasonable result with the GA. We execute the GA also with 900 generations to see if the results improve significantly by using 50 percent more generations.

We use the synthetic scene shown in figure 6.4 (left). It consists of a mesh of three connected planes at 90 degrees (1323 vertices & 2400 triangles). Two planes are

(1) GA with 600 generations results

	Mean	Standard deviation
Constraint satisfaction	0.0000442280	0.0002837807
Residual error	0.0563317010	0.0345468581

(2) GA with 900 generations results

	Mean	Standard deviation
Constraint satisfaction	0.0000401600	0.0002062299
Residual error	0.0539431480	0.0337839422

(3) Simplex results

	Mean	Standard deviation
Constraint satisfaction	0.000112920	0.000440671
Residual error	0.028951924	0.005103937

(4) Simplex with random starting point results

	Mean	Standard deviation
Constraint satisfaction	4.5975294200	20.818031398
Residual error	38.217655213	88.324896447

Table 6.2: We show here the mean and standard deviation of the constraint satisfaction and residual error for the GA with 600 generations (1) and 900 generations (2) and for the Downhill Simplex with known starting point (3) and random starting point (4).

parallel. A varying amount of Gaussian-distributed 3D noise with a variance of 0.004 is added to the vertices. The planes are constrained by two orthogonal and one parallel constraints. We are only interested in the planes of this model. Thus, the model is represented by a parameter vector of 9 floating point numbers (3 per plane).

We compare the constraint satisfaction and the residual error of the two algorithms in our evaluation. Together, these values make up the result of the evaluation function. The model describtion of the synthetic scene is optimised taking the constraints into

account 1000 times with both algorithms. Each of the 1000 experiments gives us the constraint satisfaction and the residual error of the optimised model describtion. We then calculate the mean and the standard deviation of the constraint satisfaction and the residual error over all 1000 experiments (see table 6.2). The results show us that we can obtain good results (*i.e.* low mean) with the GA (1) and the Simplex algorithm (3). Both of them reach a point (perhaps a local optimum) close to the global optimum (*i.e.* the evaluation is close to zero). It is hard to say which exact point is the true global optimum, because of the added noise. Some points are worth noting. First, the Simplex algorithm gives us a better fit than the GA. In contrast, the results of the GA satisfy the constraints better. However, the differences of the constraint satisfaction are on a much smaller scale. Overall, the Downhill Simplex algorithm ends with a much smaller evaluation function value (*i.e.* constraint satisfaction plus residual error). The optimisation results are fairly stable (*i.e.* standard deviation is low). It makes very little difference if we increase the number of generations for the GA (2). We also started the Simplex algorithm experiment with random starting points for the initial simplex (4). Neither the mean constraint satisfaction nor the mean residual error are good. The results differ widely, from very good (where the random starting point was close to the optimum) to very bad. These results have large standard deviations. The last experiment shows us that the performance of the Simplex algorithm depends very much on the starting point.

Like all optimisation methods that use function evaluation, the GA and the Downhill Simplex do not have the reputation of working computationally very efficiently. Furthermore, as the size of the parameter vector increases, the search space becomes more complex, making it more difficult for the optimisation technique to find the optima. We conducted an experiment to measure how the parameter vector size impacts the optimisation performance and computational speed of the algorithms. For this purpose we optimise models with varying parameter vector sizes. This means we optimise the model parameter for a varying number of model features without taking any constraints into account. We use the 10 lines of the synthetic scene used previously (see the right image of figure 6.4). We start by optimising the model parameters for 1 line (5 floating point numbers). The number of the lines for the following experiments is

GA results

Lines	Mean Evaluation	Standard deviation	Speed (sec)
1	0.000624478	0.001943015	4.597888
2	0.001728209	0.005351307	7.543860
3	0.002715885	0.008297902	9.959491
4	0.003763068	0.011713442	12.417552
5	0.005258814	0.016686222	15.169024
6	0.005426512	0.017257465	18.029867
7	0.004828626	0.014940796	21.518988
8	0.006187627	0.018935140	25.180380
9	0.007419067	0.023067047	30.150017
10	0.009379940	0.028787555	33.459297

Simplex results

Lines	Mean Evaluation	Standard deviation	Speed (sec)
1	0.000099148	0.000297444	0.000915
2	0.000216834	0.000650504	0.004161
3	0.000420177	0.001260533	0.009652
4	0.000528768	0.001586422	0.021894
5	0.000659733	0.001982534	0.024910
6	0.000781775	0.002347697	0.049078
7	0.000913392	0.002743686	0.063483
8	0.001102382	0.003314012	0.103332
9	0.001252562	0.003771471	0.113530
10	0.001338200	0.004019501	0.168653

Table 6.3: The table shows the evaluation (mean & standard deviation) and the execution time (in seconds) for optimising models with 1 (5 floats) up to 10 lines (50 floats) for the GA (top) and the Downhill Simplex Algorithm (bottom).

then increased by 1 until we reach 10 lines (50 floating point numbers). For each number of lines the mean and the standard deviation is calculated with the results from 100 repeated optimisations. The GA runs for a fixed number of 4000 generations.

The Simplex algorithm performs best for all parameter vector sizes (see table 6.3 bottom). It is able to locate the optima very precisely in a timely fashion. On the other hand, the GA finds a good fit, but fails to refine the parameter vector much further (see table 6.3 top). The result of the algorithms vary more (*i.e.* higher standard deviation) than in the previous experiment. The execution time for the Downhill Simplex is well under 1 second in all our experiments. In contrast, the execution time of the GA is much higher. One could reduce the time the GA takes by decreasing the number of generations. However, especially with more lines to optimise, it becomes necessary to have a high number of generations to reach a reasonable result.

The result of the experiments is that we can obtain a good optimisation (at least close to the global optimum) with both algorithms. The Simplex algorithm is able to drive the evaluation function much further down. However, the Simplex algorithm depends very much on a good starting point. The optimisation results are fairly stable. As the size of the parameter vector increases the GA becomes relatively more time-consuming.

6.4 Discussion

This chapter discussed the model optimisation which produces a model description satisfying the previously-selected constraints. We used two quasi-global optimisation algorithms (GA & Downhill Simplex) to optimise the 3D model. We evaluated both algorithms on synthetic data. The optimised model description was used to flatten planes, straighten lines and, in this process, to apply the constraints. Surface ripples on planes and jitter on lines were removed. Furthermore, parallelism and orthogonality between planes and lines were enforced.

We represent the planes by azimuth and elevation angles and the distances to the origin (3 floating point numbers), which is the minimal representation. The lines are represented by a 3D point and the azimuth and elevation angles (5 floating point num-

bers). It is even possible to reduce the number of line parameters further by using the 3D point as a point on a reference plane in the origin (2 floating point numbers) and the azimuth and elevation angles (4 floating point numbers in total).

The GA reaches the global optimum region but has trouble reaching the exact optimum location. One could utilise the GA to get close to the optimal region. We can then switch to the Downhill Simplex with the optimal region serving as the starting region of the simplex. The Simplex algorithm refines the model parameters. However, we have estimated the model parameters in Chapter 4. Therefore, we can already start the optimisation with a position close to the optimum in our multi-dimensional search space. We do not have to search the whole parameter space with a global optimisation technique. We can use a local optimisation technique which crawls in the right direction. We have shown that a technique such as the Downhill Simplex optimisation that is local, yet insensitive to local optima, is sufficient to obtain a good choice of model parameters and even outperforms the GA at the final exploration of the search space.

Simple scenes including only a small number of features can be represented by a small model description (parameter vector). However, more complex scenes might include a large number of planes and an even larger number of architectural linear elements (lines). Typical architectural models can include several hundred lines (see figure 3.2) not only at plane intersections but also at windows, doors and other architectural structures. The size of the parameter vector for such a model becomes increasing large. A GA becomes very impractical for such a model. The GA performs well computationally if we optimise the model parameters for a model with only a small number of features (*e.g.* 5 planes in [CFD02b]), but works much more slowly when we optimise a model with several hundred lines (several hundred parameters). One could overcome this problem by implementing a parallel GA which can optimise the model within a computer cluster in a timely fashion. However, on a normal configuration (a single computer) the Simplex algorithm can handle this kind of scene in a way that is computationally far more efficient [CFD02a], but requires a good initial estimation of the feature parameters.

The optimisation is applied by orthogonally projecting the points onto their features. Other research has proposed applying the changes by re-triangulation [RDSG01].

Re-triangulation performs a new triangulation of the mesh around the planes and lines. This method preserves structural information. It is useful if points from one line are likely to be projected onto another line (that would destroy the first line).

Chapter 7

Applying architectural constraints

A framework for application of constraints in the reconstruction process was described in the previous chapters. In this chapter the application of the techniques is demonstrated with different kinds of 3D models.

The proposed framework is general. It is independent of the way in which the 3D model was created and of model properties such as variance of the triangle size. It has been applied to several triangulated models that were constructed from different input sources (*e.g.* range or intensity data) and with different techniques (*e.g.* different triangulation algorithms).

We will present resulting constrained models created from synthetic data, from miniature scenes of factories and from models of real architectural scenes. The use of synthetic models enables us to evaluate the framework with varying levels of noise. We use the Downhill Simplex for the experiments here, because of the advantages of this optimisation technique demonstrated in the last chapter, especially if the parameter vector becomes very large. The experiments are conducted on a state-of-the-art single PC with a 2GHz processor and 1GB memory.

7.1 Synthetic scene

For the first experiments, we use synthetic data. Synthetic models offer several advantages over models reconstructed from real scenes. A synthetic dataset allows evalua-

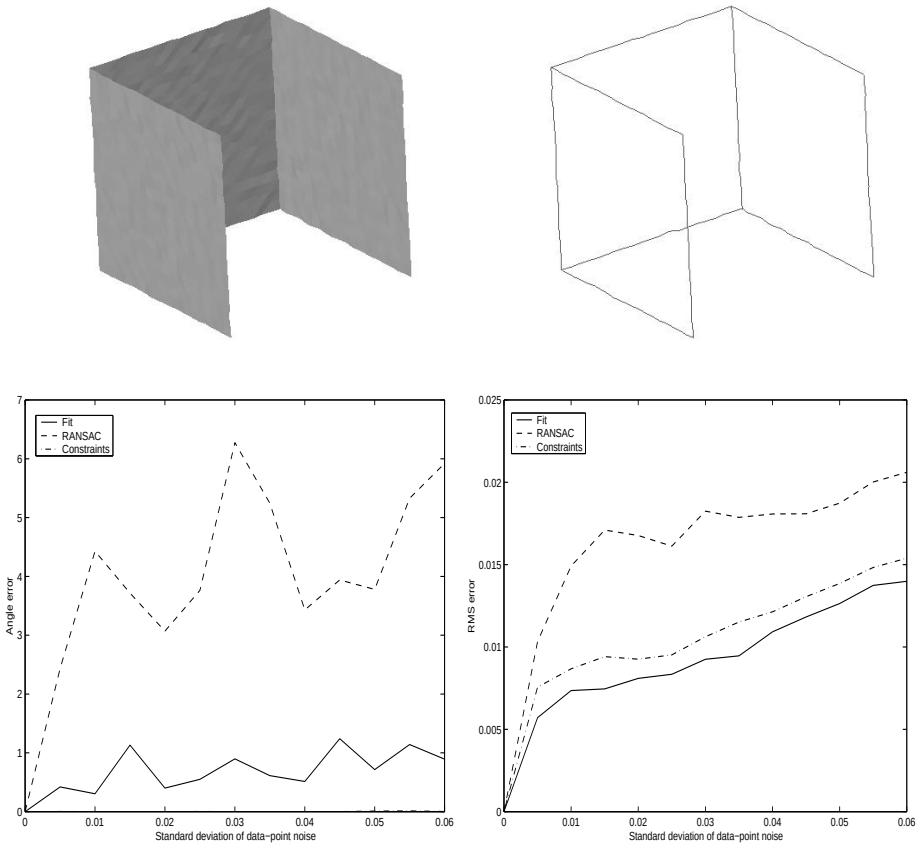

Figure 7.1: The top images show the synthetic scene with added noise (left) and as extracted lines (right). The bottom graphs show the experimental results. The left graph shows the angle error in degrees versus the noise level. The Constraints line lies very close on the noise axis. The graph on the right shows the mean squared residual versus the noise.

tion of the performance against known ground truth. Furthermore, we can see how the model parameter estimation copes with controlled added noise in the 3D data. We may analyse the model descriptions provided by the feature extraction (RANSAC: dash line) and the model optimisation, by taking either the fit and the constraints into account jointly (Constraints: dash-dot line) or by taking the fit into account on its own (Fit: solid line). The synthetic triangular model used for the evaluation consists of

a perfect mesh with uniformly distributed vertices. A varying amount of Gaussian-distributed 3D noise is added to the vertices. The mesh consists of three planes at 90 degrees with 1323 vertices and 2400 triangles (see figure 7.1 top). Two planes are parallel.

When the constrained reconstruction procedure is run on this scene, 3 planes and 10 lines are extracted. The 10 lines are successfully grouped into 3 clusters of parallel lines from which the principal directions are calculated. The parameter vector which describes the model consists of 45 floating point variables (9 for 3 planes, 30 for 10 line starting points and 6 for 3 cluster orientations). The planes are constrained by three constraints. Additionally, three orthogonality constraints from the three principal directions are applied.

The graphs in figure 7.1 show how the application of constraints improves the reconstruction. The first graph (see figure 7.1 bottom left) shows the angle error from the initial feature extraction (dash line), improving the fit (solid line) and application of the constraints (dash-dot line, bottom near noise level axis). The angle error is the variation of the angles between the plane and line orientations from the optima (*i.e.* truly orthogonal or parallel). A low angle error means a good constraint satisfaction. Improving the fit induces projecting vertices onto the closest planes, but does not use any constraints in the evaluation function (Eval=$SR(\vec{p})$). Effectively, this minimises the residual error of the model. The angle error from feature extraction varies strongly as a result of the random nature of RANSAC. Specifically, the orientation of the three principal directions varies widely, because fewer points are used to fit the lines and estimate the three directions. Improving the fit using all data points from the features gives better results. Finally, using the constraints gives an angle error close to zero. The second graph (see figure 7.1 bottom right) shows the mean squared residual after feature extraction (dash line), improving the fit (solid line) and constraining the model (dash-dot line). The parameters obtained from feature extraction show the biggest error. The mean residuals from improving the fit and from applying the constraints are close together and are both significantly below the dashed feature extraction line. The two graphs show that applying constraints (dash-dot line) improves the orientation of planes and lines without significantly worsening the fit from a pure fit (solid line in

right graph).

7.2 Architectural miniature scenes

We now apply the constraints to reconstructed models of miniature factory scenes. The models have dimensions of roughly 40x20x20 cm and include planar surfaces and much linear structure in form of pipes. The miniature models were scanned with a very precise orthogonal range scanner. The scanner offers a maximum resolution of 0.1 mm steps between scan points over a depth range of approximately 10 cm. Experiments have indicated that the range measurement error is approximately 15 microns.

7.2.1 Factory scene 1

The first factory model includes 3 planes and some small vertical pipes (see figure 7.2). It was initially reconstructed by an edge-preserving Marching Triangle algorithm [MF02], to include 19686 vertices and 37302 triangles. This model and the other models reconstructed from real scenes have been decimated [LT98] to speed up the RANSAC based feature extraction. The decimated model has 13834 vertices and 26260 triangles.

The planar surfaces are fairly smooth in the miniature models thanks to the accuracy of the laser range finder. Therefore, we can use a low distance threshold (0.38) for the plane extraction. A unique labelling of the planes in the scene is not possible, since it does not provide enough planes to form a house. Nevertheless, we can assign three constraints to enforce near-regularities in the orientations of the planes. The edges are fairly straight due to the constrained Marching Triangle Algorithm. We can therefore use a low distance threshold (0.014) for the line extraction here as well. We extract 3 planes and 21 lines (see figure 7.2). Parallel lines are grouped into 5 clusters and the three principal directions (made up from 18 lines) are identified. The parameter vector consists of 82 variables (9 for the 3 planes, 63 for the 21 line starting points and 10 for the 5 clustered line orientations).

Six constraints are applied to the data. Three constraints enforce the plane orientations and 3 constraints enforce the three principal directions. The runtime of the

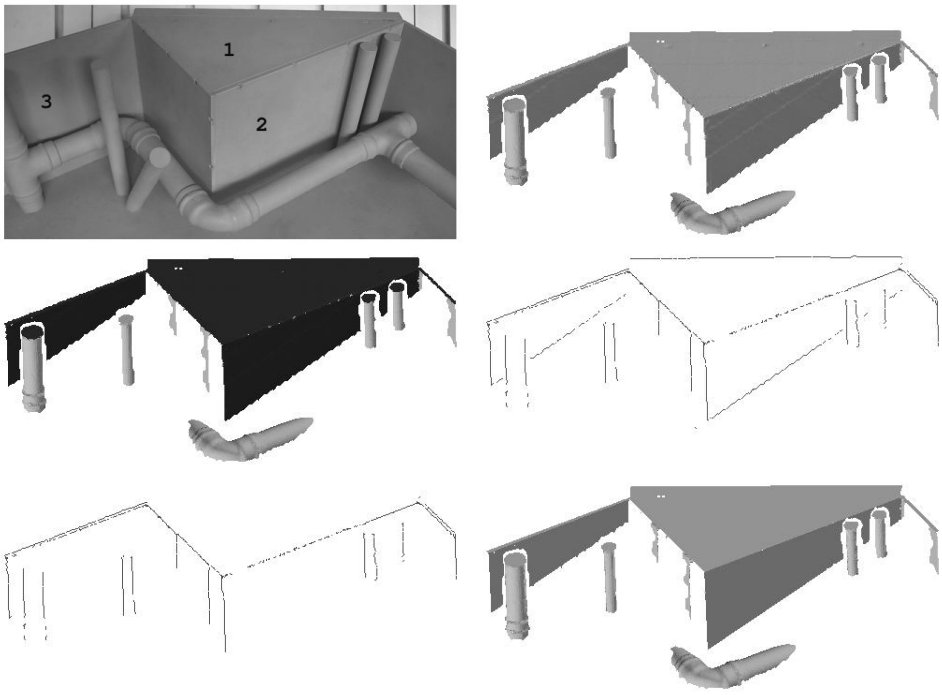

Figure 7.2: The images show factory scene 1. On the top are the real scene (left) and the triangulated model (right). The middle images display the 3 extracted planes in darker colour (left) and 21 extracted lines (right). The bottom row shows the three principal directions (left) and the constrained model (right).

Simplex Downhill optimisation is 12 seconds. The average movement of the moved vertices in the model as a fraction of the model diameter is 0.11%. The model diameter here is 2, since the model is mapped into a unit sphere. Figure 7.3 shows close-up views of the scene. One sees there how small surface irregularities such as the stripes on the planar surfaces are removed (see top images) and lines are straightened (see top & middle images). The small holes in the triangulated model on the left and along the main roof edge (top images) come from the triangulation algorithm used to create the original model.

The angles between the three planes in the scene are constrained to be orthogonal (between plane 1 & 2 and 1 & 3) or parallel (between plane 2 & 3). Table 7.1 shows the extracted angles obtained from feature extraction, improving the fit (as we have done

Planes	Extraction	Fit	Constrained	Tolerance	Real
1 - 2	89.824569	89.697657	89.999968	89.953537	89.95
1 - 3	89.479918	89.584133	89.999948	89.950238	90.00
2 - 3	0.351458	0.148514	0.000100	0.046388	0.00

Table 7.1: The table shows the angles between the three extracted planes (as seen in figure 7.2 top/left) of factory scene 1 from plane extraction, improving the fit, enforcing constraints, enforcing constraints with an angle tolerance of 0.05 degrees and measurements on the real model (from left to right).

with the synthetic data), application of the constraints, application of the constraints with a small allowance in form of an angle tolerance of 0.05 and the real angles[1] in the original miniature scene (ground truth). The initial model description provided by the feature extraction is already fairly good due to the precision of the range finder. Improving the fit (minimising the residual error) does not correct the orientation very much. After application of constraints between the planes, the constraints differ less than 0.001 of a degree from the perfect orientation, although, the angle between plane 1 and 2 is already slightly too tightly constrained. We also apply constraints with a small angle tolerance for the plane constraints of 0.05 degrees to allow small variations of the orientations. The tolerance of the angle constraints preserves small misalignments in the data. However, the orientations of the other planes then move away from their perfect orientations.

7.2.2 Factory scene 2

The second factory model is a reconstruction of the same scene viewed from a different viewpoint and including additional objects. It includes three planes, a variety of small pipes and a big cylindrical surface on the bottom (see figure 7.4). The reconstructed triangulated model was reconstructed by simply connecting the points of the subsampled single 2.5D range image. This results in much jitter at the edges (see figure 7.3 bottom for an example). The original model included 17443 vertices and 32844 triangles. It

[1] measured in a mechanical workshop

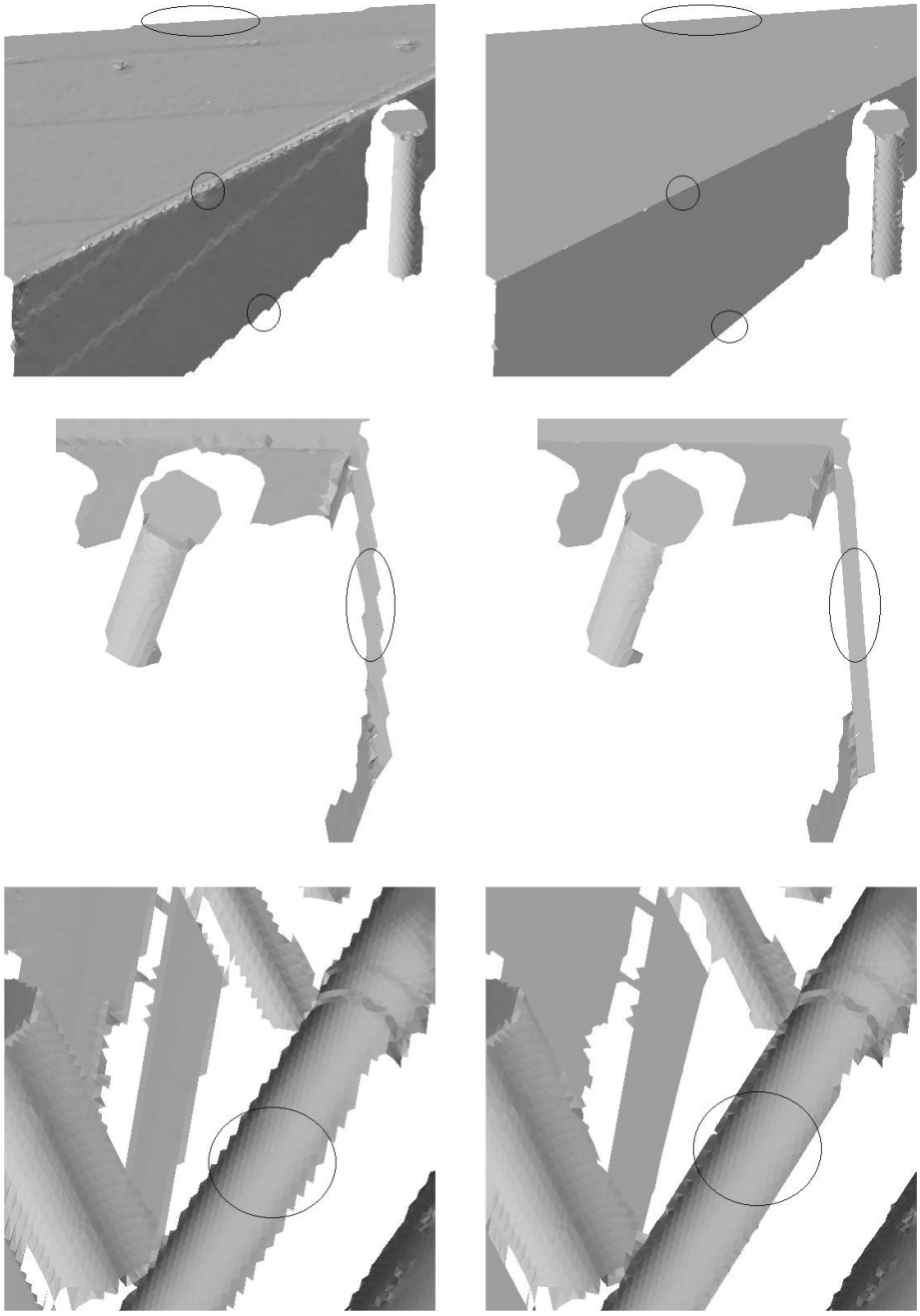

Figure 7.3: These images show close-up views of the factory scenes before (left) and after (right) application of constraints. Row 1 & 2 show scene 1 and row 3 shows scene 2. Irregularities on planes are removed (top images) and lines are straightened (all images). Straightened lines are most easily seen at the circled regions.

was then decimated [LT98] to 13982 vertices and 26002 triangles.

The planar surfaces are fairly smooth in this model as well, therefore, we can again use a low distance threshold (0.44). The planes of the scene do not form a house, so they cannot be labelled as walls, ground floor and so on. The edges include much jitter, because of the simplistic triangulation method. To extract lines we need to use a higher distance threshold (0.018) here. We extract 3 planes and 54 lines (see figure 7.4). The lines are grouped together to form 17 clusters and the three principal directions (including 22 lines) are identified. The parameter vector includes 9 floating point variables for 3 planes, 162 floating point variables for 54 line starting points and 34 floating point variables for 17 cluster orientations (205 variables in total).

The optimisation takes 6 constraints into account, 3 for constraining the 3 planes and 3 for constraining the three principal directions. The optimisation takes 21 seconds. It takes considerable longer than the optimisation for the first factory scene, since the model description is much larger (82 vs. 205 floating point numbers). The average movement of the projected vertices as a fraction of the model diameter is 0.30%. The movement is much larger than with the first factory model. This model is much rougher with much jitter at the edges. Thus, we have to use larger distance thresholds and the movement of the vertices becomes larger. Small surface irregularities are removed and in particular much of the jitter at the edges is removed as can be seen in the close-up view of the pipe in figure 7.3.

The angles between the three planes in the scene are constrained to be orthogonal.

Planes	Extraction	Fit	Constrained	Tolerance	Real
1 - 2	89.932329	89.819119	89.999067	89.952376	90.00
1 - 3	89.875570	89.883609	89.999538	89.981684	90.00
2 - 3	89.770098	89.827766	89.999702	89.949978	89.95

Table 7.2: The table shows the angles between the three extracted planes (as seen in figure 7.4 top/left) of factory scene 2 from plane extraction, improving the fit, enforcing constraints, enforcing constraints with an angle tolerance of 0.05 degrees and measurements on the real model (from left to right).

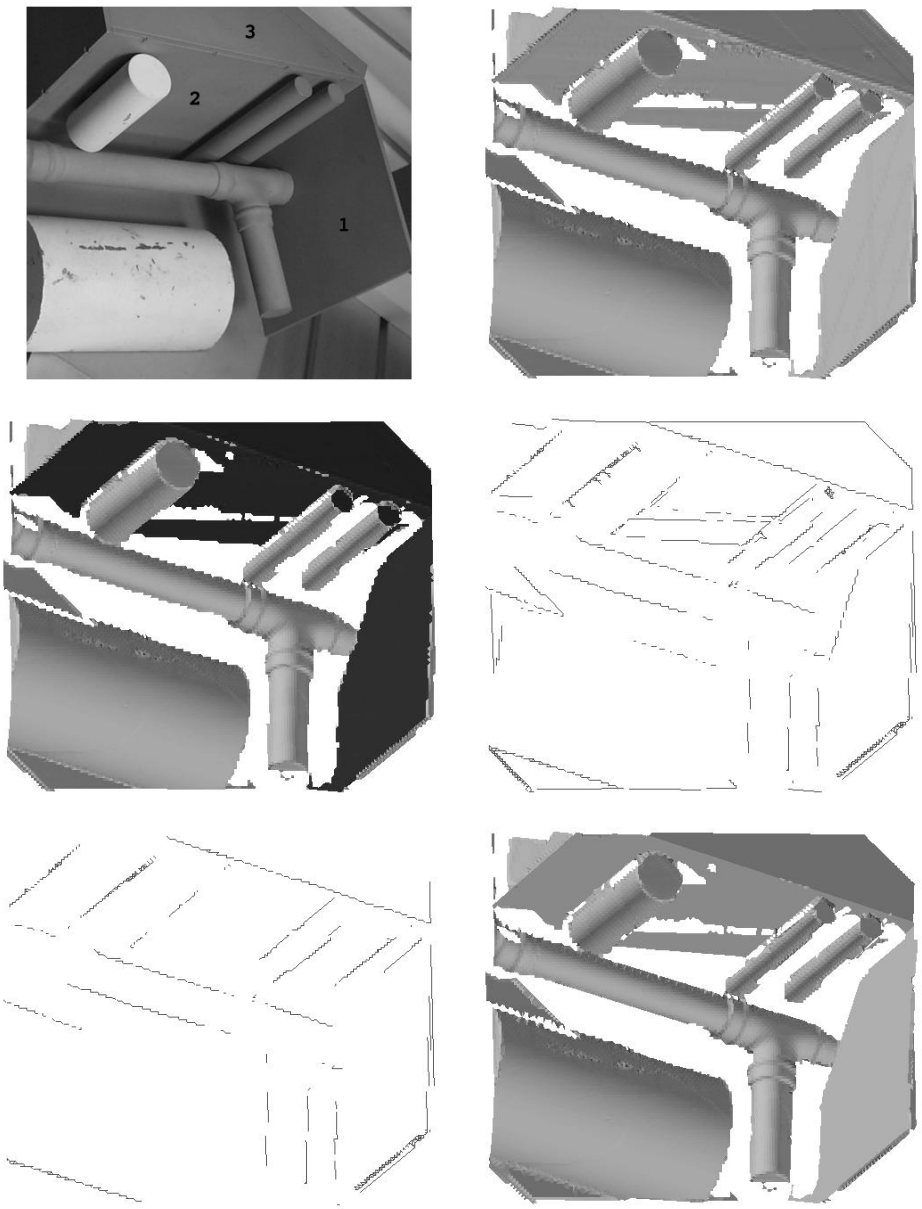

Figure 7.4: The images show factory scene 2. The top row shows the real scene and the initial triangular model. The middle row shows the 3 extracted planes (darker colour) and the 54 extracted lines. The bottom images show the three principal directions and the model with enforced constraints.

We have orthogonality constraints between between plane 1 & 2, 1 & 3 and 2 & 3. Table 7.2 shows the extracted angles obtained from feature extraction, improving the fit, application of the constraints, application of the constraints with a small allowance in form of an angle tolerance of 0.05 and the real angles[2] in the original miniature scene. The initial model description provided by feature extraction is already fairly good as it was for the first factory scene. Improving the data fit minimises the residual error of the data. But, improving the fit does not move the plane orientations in the model to the plane orientations in the real scene. We do not recover the true orientations until we apply constraints between the planes. The orientations differ less than 0.001 of a degree from the perfect orientation after optimisation. However, the angle between plane 2 and 3 is too tightly constrained. Again, to preserve the angle of 89.95 degrees we optimise the model also with an angle tolerance of 0.05 degrees. However, the orientations of the other planes again move away from their perfect orientations.

7.3 Architectural scenes

We show in the last result section how we constrain 3D models reconstructed from real architectural scenes. Models from different kinds of architectural styles are used here. The first architectural model is from Arenberg castle, an ornate 15th century mansion with late-gothic and renaissance influences. The second model is from a typical Bavarian farmhouse and the last model is from the Edinburgh Central Mosque and Islamic Centre. It is a modern building (opened 1998) with a distinctive shape.

7.3.1 Arenberg castle

The first model from a real scene is a reconstruction from Arenberg castle (in Belgium). It was reconstructed by the Catholic University of Leuven [Pol99]. The reconstruction process uses an image sequence of 20 images to construct a texture-mapped model consisting of 6292 vertices and 12263 triangles.

Five planes are extracted from the model. Three planes are labelled as wall, one as floor and one as roof plane. The roof planes in the model are partly deformed, because

[2] measured in a mechanical workshop

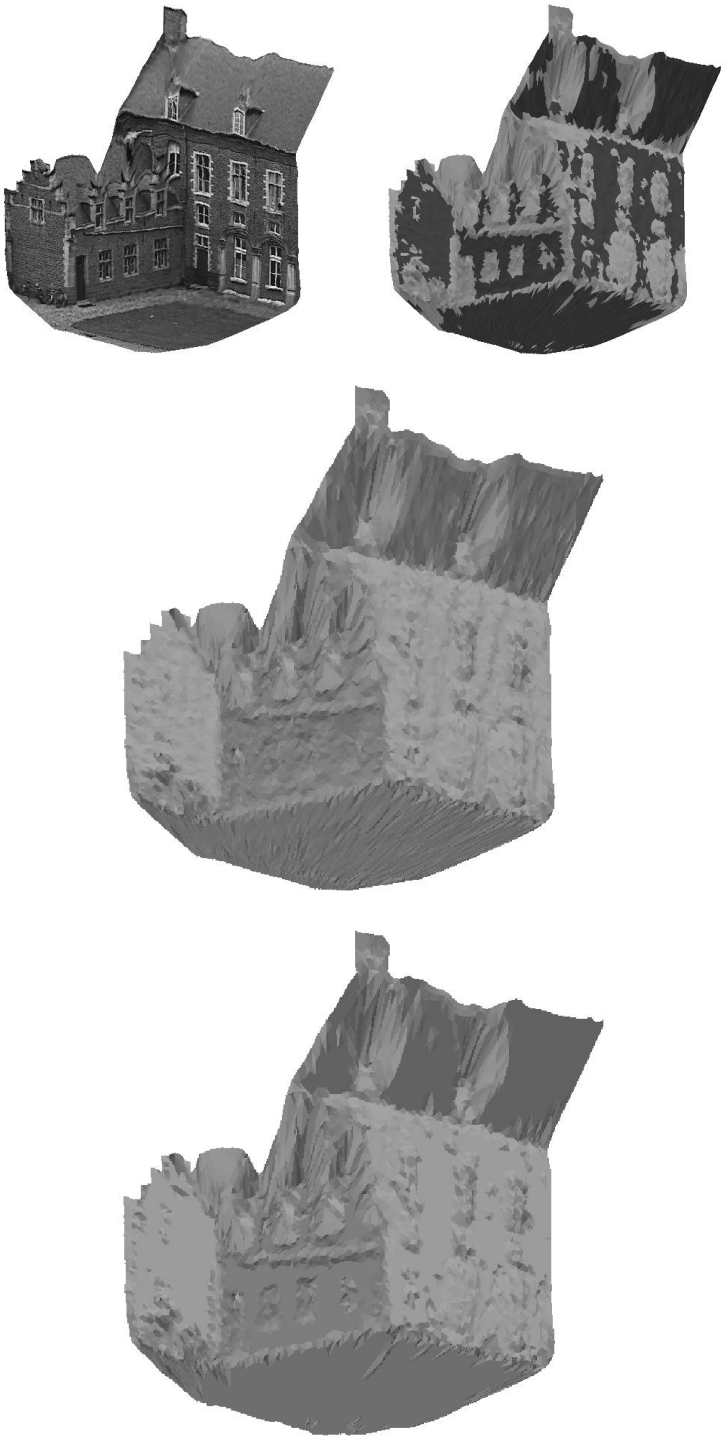

Figure 7.5: The top images show the textured model (left) and the extracted planes in darker colour (right). Below are the original solid model and the resulting model after optimisation.

of the viewpoint the image sequence was taken from. The walls and the ground show a lot of small irregularities. We need to use a rather high distance threshold (0.84) to extract the walls. Unfortunately, this scene does not include enough evidence for the extraction of lines. The model description (parameter vector) is therefore fairly small for this scene, since we only take planes into account. The vector consists of only the 15 floating point variables for the 5 planes.

The 5 extracted planes are constrained by 7 constraints. The angles between the planes vary from the optimum by 1.5 degrees on average before optimisation. The optimisation takes only 1 second, because of the small dimension of the search space (parameter vector size). After optimisation the orientations differ less than 0.01 degrees from the optimum. The average movement of the plane vertices as a fraction of the model diameter is 0.33%. The movement is fairly large in this model, because of the small irregularities and thus the large distance threshold used to extract planes. The walls and the ground on the original solid model show clearly a lot of small irregularities which are removed in the constrained model (see figure 7.5).

7.3.2 The Bavarian farmhouse

The second scene is from a Bavarian farmhouse initially reconstructed by the European Commission Joint Research Centre (JRC) [SNW+99]. The model is shown in figure 7.6. The model was reconstructed from multiple range data scans. This is a full 3D model including all sides of the building. The mesh consists of 12504 vertices and 16589 triangles.

Our plane extraction (plane distance threshold 0.7) finds 6 planes with 1856 vertices. The low number of plane vertices in comparison to the total number of vertices (only a tenth) results from the fact that the planes consist of relatively few big triangles and that model details like windows consist of many small triangles. The plane extraction preserves features like the windows and doors (see Figure 7.6). Four planes are labelled as walls (front, back and one on each side) and 2 planes form the roof. The line extraction identifies 63 lines (line distance threshold 0.01) and they are grouped together in 44 clusters. All lines in any one cluster are considered to be parallel to each other. The lines in this scene have a great variety of orientations, hence the large

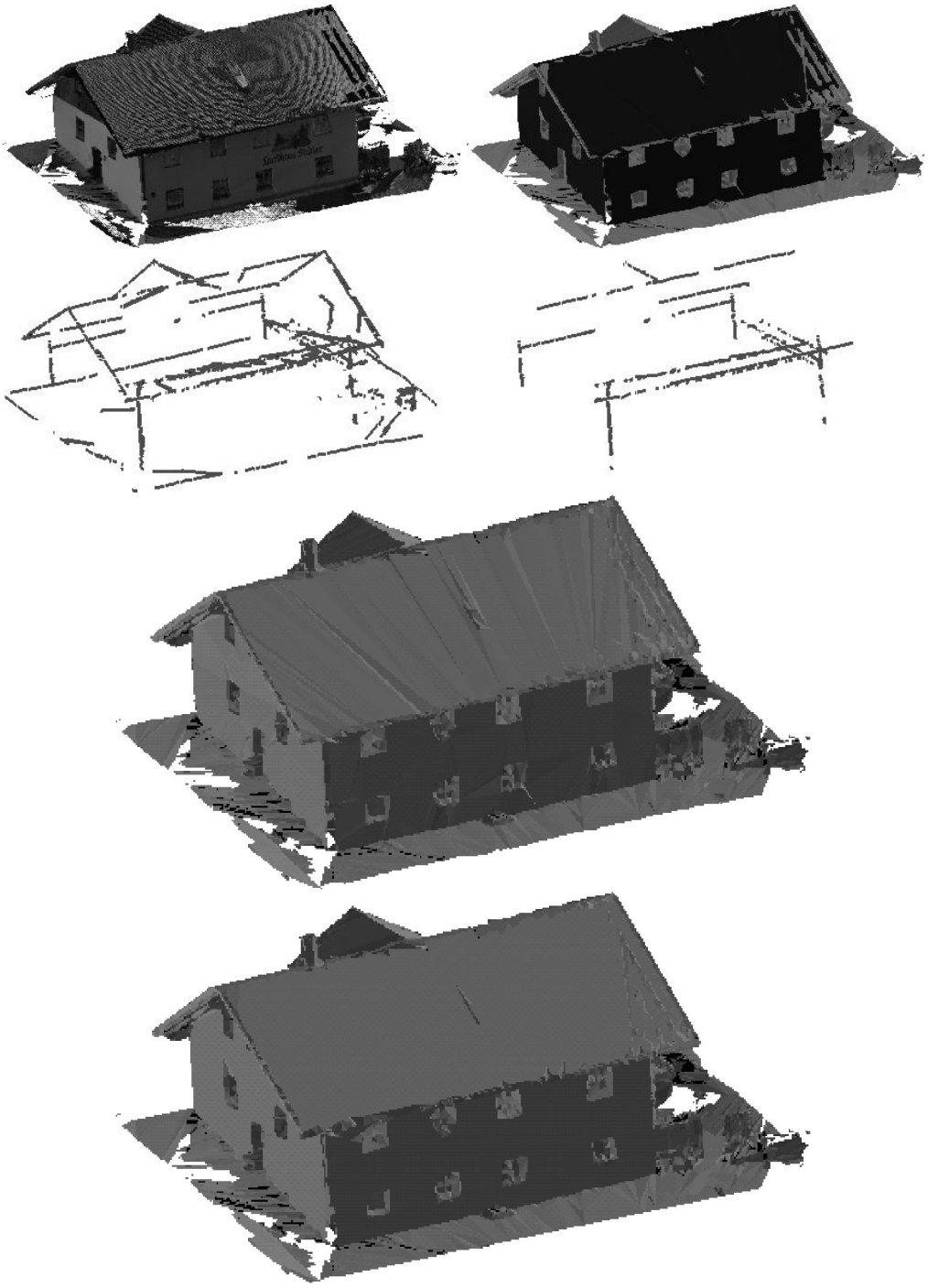

Figure 7.6: The top images show the textured model (left) and the extracted planes in darker colour (right). The middle images display the extracted lines (left) and the three principal directions (right). Below are the original solid model and the resulting model after optimisation.

Figure 7.7: A close view of one wall of the farmhouse. On the left is the original unconstrained model. Surface ripples between the windows are most easily seen in the circled areas. On the right is the optimised model with fewer irregularities.

number of clusters. The parameter vector consists of 295 variables (18 for the 6 planes, 189 for the 63 line starting points and 88 for the 44 cluster orientations).

The initial parameters obtained from the feature extraction give us angle errors that are no more than 1.3 degrees off. The angle errors of the plane and line orientation vary from the optimum by 0.4 and 0.7 degrees on average. The planes are constrained by 10 constraints and the three principal directions by three constraints. The optimisation with the Downhill Simplex algorithm takes 31 seconds. After optimisation all angle errors differ less than 0.01 degrees from the optimum. The result in figure 7.6 shows the model with irregularities removed and planes and lines aligned. The average movement of the moved vertices is 0.21% of the model diameter. The improvement of the model is visibly shown in a close-up view of a wall in figure 7.7. The original solid model shows small edges on the walls, whereas the optimised model has these edges projected onto the wall.

7.3.3 Edinburgh Central Mosque

The last scene we use to demonstrate our constrained reconstruction techniques is the Edinburgh Central Mosque and Islamic Centre. Models of the inside and the outside were reconstructed from multiple registered range data scans by the Marching Triangle algorithm [MF02]. The models are full 3D models including data from all four sides of the building. The outside model does not include the roof. It is unlike the previous models in the way that it consists of much more structure (*i.e.* planes and lines). The model description is much larger, much more constraints are applied and thus the model optimisation is more difficult.

7.3.3.1 Inside

The model of the inside of the mosque presents the main prayer hall. It is a very large model with 184245 vertices and 364070 triangles. To make it computationally easier to extract features, the model was decimated [LT98] to 33364 vertices and 65708 triangles (see top of figure 7.8).

The main prayer hall of the mosque consists of a large high hall with many columns and a flat ceiling with different height levels. We extract 22 planar surfaces (plane distance threshold 0.5) and 112 lines (line distance threshold 0.008) from the data (see middle of figure 7.8). We disabled the connectivity checking during the plane extraction, because the cluster analysis was computationally not feasible for this large model. Creation of the cluster hierarchy for planes with several thousand triangles is computational not reasonable on our evaluation PC. So, disconnected planes are not split into several smaller planes. Many of the lines are on the columns and between the displaced walls on the back of the building. The lines are grouped into 8 clusters of parallel lines. Most of the lines in the scene are part of the three principal directions. The 3 clusters that form the principal directions include 107 of the 112 lines in the scene. The model description (parameter vector) consists of 418 variables (66 for the 22 planar surfaces, 336 for the 112 line starting points and 16 for the 8 clustered line orientations).

Geometrical constraints are assigned to pairs of nearly parallel/orthogonal planes. However, taking the high number of planes in the model (unlike previous models) into

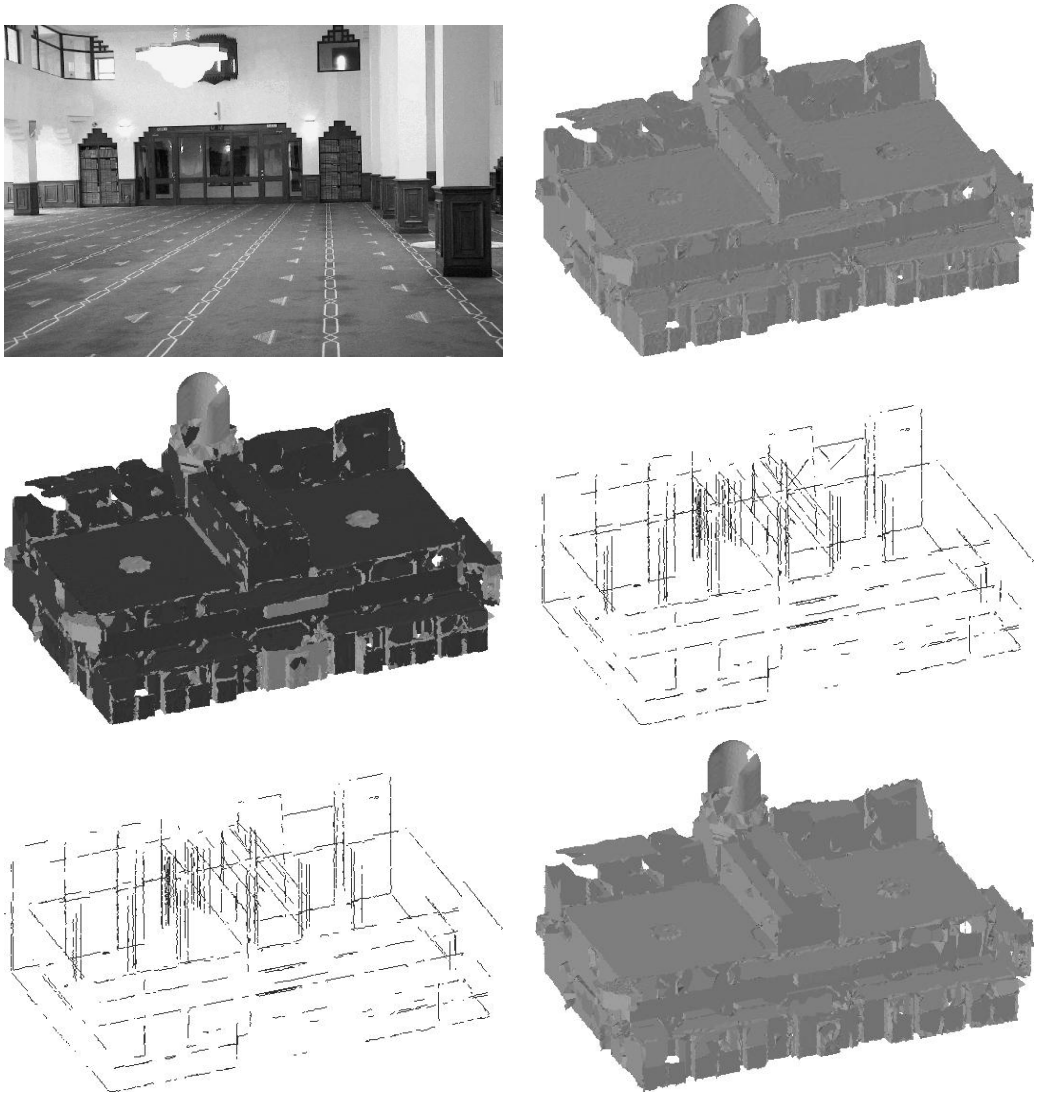

Figure 7.8: On the top are a photograph of the prayer hall (left) and the solid model (right) of the inside of the mosque. The view here is of the backside of the inward facing surfaces. The middle row shows the extracted planes in darker colour (left) and lines (right). At the bottom are the lines of the three principal directions (left) and the optimised model with flattened, straightened and aligned planes and lines (right).

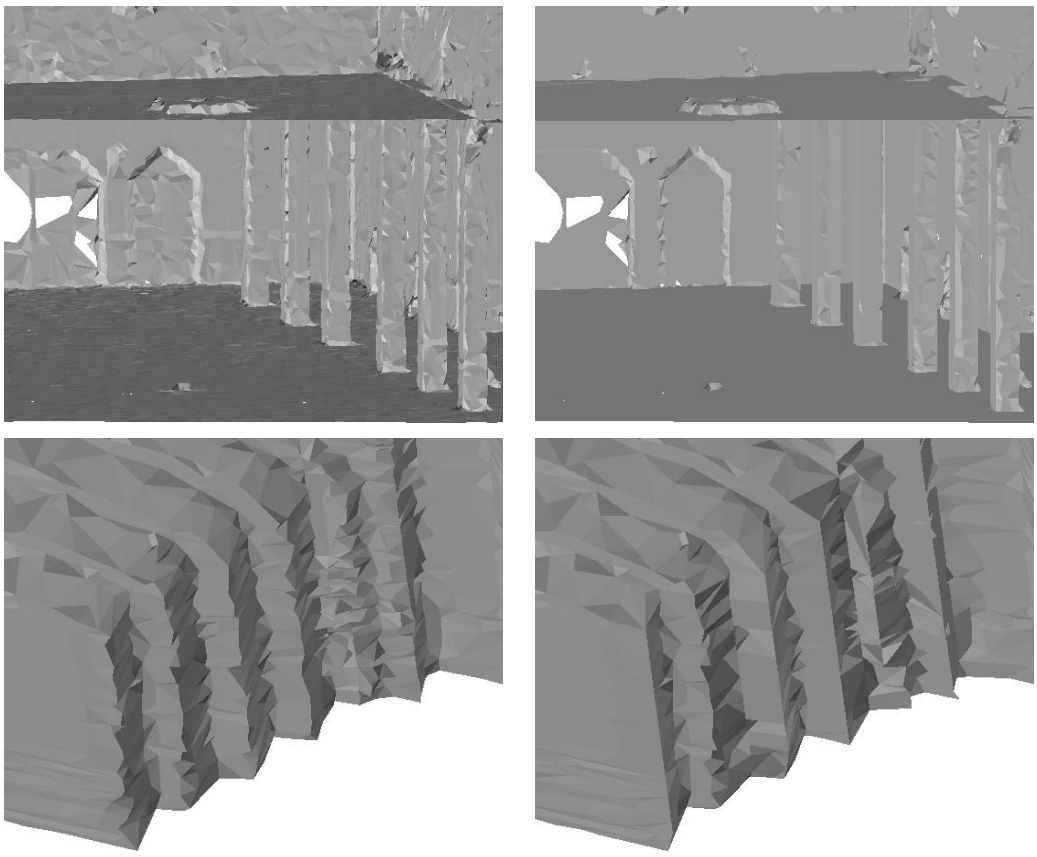

Figure 7.9: These images show close-up views of the indoor mosque models before (left) and after (right) optimisation. The top images show how the planes are flattened in the inside of the mosque. The bottom images display the back of the mosque. One sees how lines are straightened.

account, one might get better optimisation results by using the cluster technique we have used for line grouping. The size of the parameter vector would be slightly reduced (*i.e.* parallel planes have the same variables for the orientation) and the number of the constraints would be much smaller. However, here each plane is allocated its own orientation and 231 constraints are used to enforce the orientations of the 22 planes. Additionally, 3 constraints enforce the three principal directions. The optimisation step takes 172 seconds. The average movement of the moved vertices as a fraction of the model diameter is 0.17%.

After optimisation all angle errors differ less than 0.1 degrees from the optimum.

The constraint satisfaction is considerably smaller than it was with the previous model data. The large number of constraints affects the satisfaction of the constraints. If we use only 10 constraints, they are all very well satisfied (0.01 degrees from the optimum). Small bumps and edges on the planes are removed (see the top images of figure 7.9) and lines are straightened. The bottom images in figure 7.9 show a close-up view of the displaced walls at the back of the mosque. Lines have been extracted at the fold edges between the walls. One sees how they have been straightened in the optimisation process. However, a few lines we can identify as human observers were missed in the automatic extraction process. The line extraction was not able to identify all lines at the folds, because some of them were too distorted. So, not all possible lines were constrained.

7.3.3.2 Outside

The model of the outside of the mosque consists of 78577 vertices and 154331 triangles. We decimate [LT98] the model to 25783 vertices and 49011 triangles (see figure 7.6).

The outside of the mosque includes many planar surfaces especially at the corners of the building. We extract here even more planes than in the indoor model. 33 planes (plane distance threshold 0.5) and 87 lines (line distance threshold 0.009) are extracted (see middle of figure 7.10). As with the model of the prayer hall, the connectivity constraint during plane extraction is disabled. We treat disconnected planes as one big plane. The scene includes rough surfaces beside the building. To avoid extracting lines at those surfaces unnecessarily, we restrict the set of initial edges for the line extraction to jump edges and roof edges that lie at the borders of the extracted planes (see section 4.3.1). Therefore, the extracted lines reflect the silhouettes of the planes. The lines are grouped into 7 clusters of parallel lines. Eighty of the lines are in the three principal directions. The model description (parameter vector) consists of 374 variables (99 for the 33 planar surfaces, 261 for the 87 line starting points and 14 for the 7 clustered line orientations). The model description is smaller than the one for the indoor model (418 variables) despite the larger number of planes, because the number of lines is much smaller here (87 vs. 112 lines).

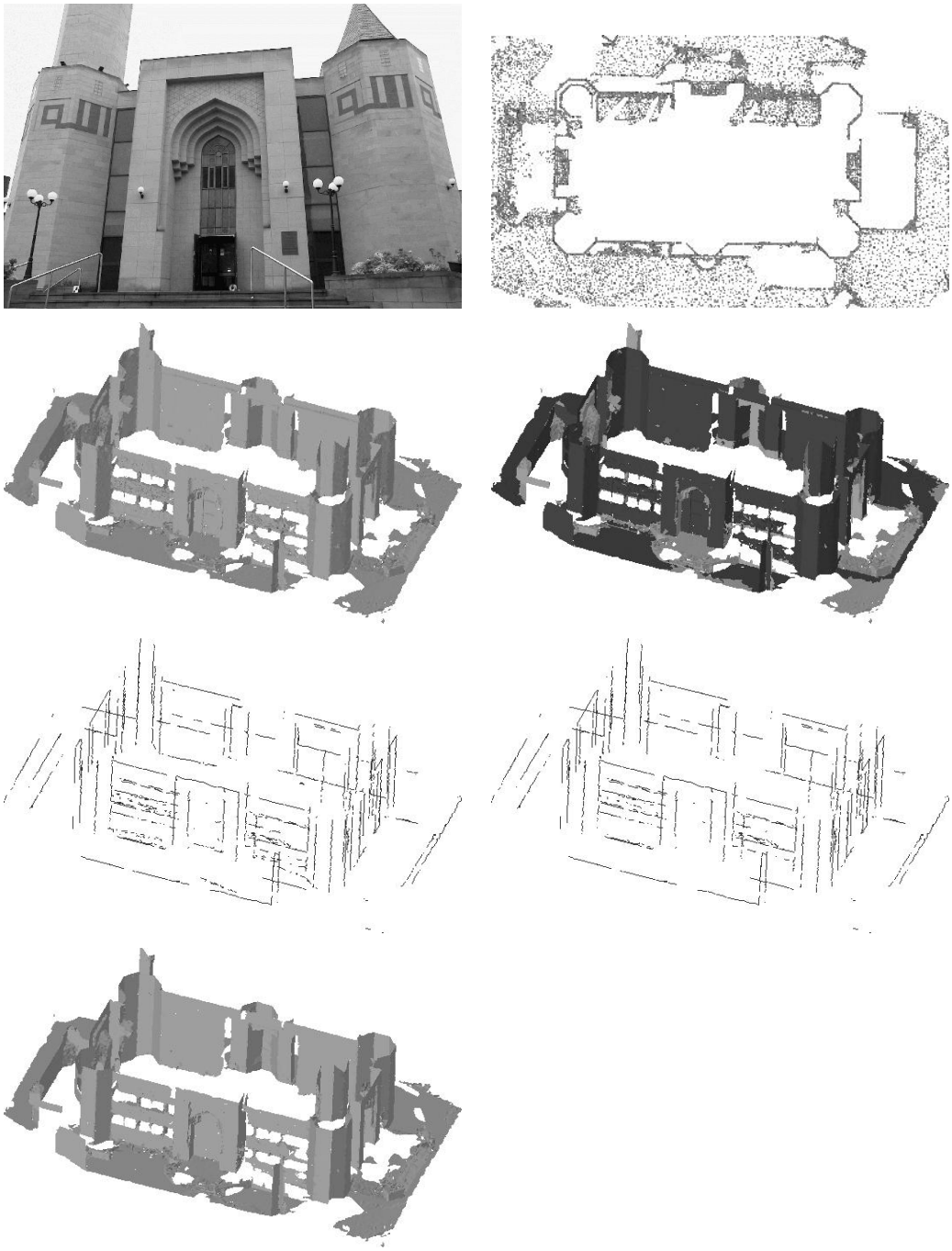

Figure 7.10: At the top is a photograph of the front of the mosque (left) and a view of the model vertices from the top (right). The minaret is in the top left of the images. Below are the solid model (left) and the extracted planes in darker colour (right). The minaret is here on the right in the foreground. Below are the extracted lines (left) and the three principal directions (right). At the bottom is the optimised model.

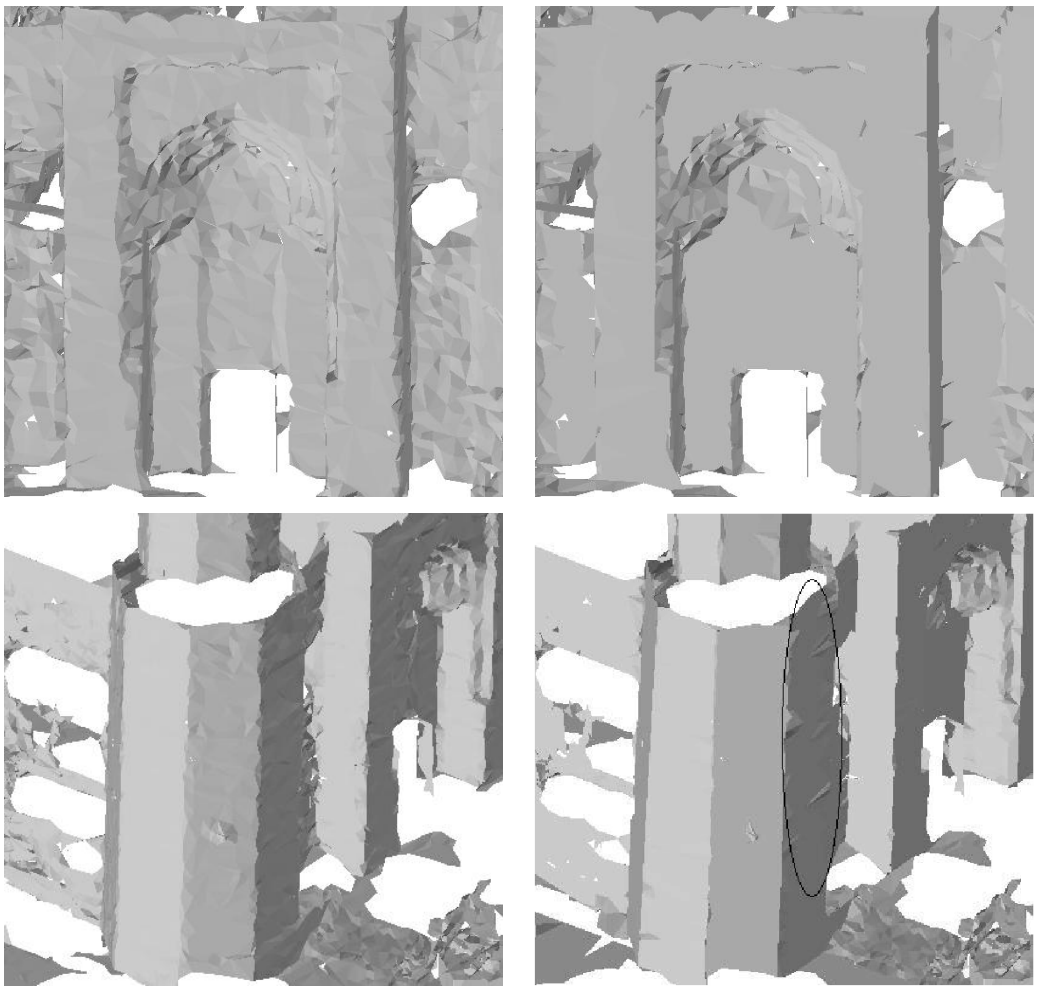

Figure 7.11: These images show close-up views of the outdoor mosque models before (left) and after (right) optimisation. The images at the top show the entrance of the mosque. One sees how the planes get flattened and the lines straightened. The bottom images show how the projection of points on the border between two planes at 45 degrees interferes on one plane. The effect is most easily see on the circled plane.

We assign 274 constraints to enforce the orientations of the 33 planes in the model. The ratio of constraints to planes in this model (274:33 $\approx$ 8.30) in comparison with the ratio in the indoor model (231:22 $\approx$ 10.50) is smaller because many of the walls are at 45 degrees to each other. This means no constraints are assigned to those pairs. Three orthogonality constraints are used to enforce the orientations of the the three principal directions. The optimisation step takes 97 seconds. The average movement

of the projected vertices as a fraction of the model diameter is 0.25%. As with the indoor scene, the constraint satisfaction is not as good as with the previous smaller models that use fewer constraints. The top images of figure 7.11 show the entrance of the mosque. The planes are flattened and the lines are straightened. However, small depth structures on the wooden door are removed during plane flattening, because the magnitude of these structures is smaller than the distance threshold we used to extract the planes. The walls at the corners of the mosque have the shape of a octagonal prism. Unlike in the previous models, the walls are not orthogonal to each other. They are aligned at a 45 degree angle. The double projection of vertices at the border of two walls at 45 degrees can lead to disturbances to the plane onto which points had been projected first (see the bottom images of figure 7.11). The left image displays the base of the minaret in the original 3D model. It has the shape of a octagonal prism. There are small irregularities on the walls of the minaret. The optimised model on the right has the walls flattened. However, the wall on the right shows small edges introduced by the projection of the walls beside it.

7.4 Discussion

In this chapter we have demonstrated our proposed technique on several 3D models from both synthetic and real scenes. We have used models from real data with differing complexity. Table 7.3 shows the different models and their complexities. As the model complexity (*i.e.* number of planes and lines) increases and thus the size of the parameter vector ('Parameters') increases, the execution time of the optimisation ('Time') increases. The number of the applied constraints ('Constraints') is the sum of the applied plane and line constraints. We used these models to demonstrate the extraction of planes and lines, the labelling of planes, clustering of parallel lines, selection of constraints, the optimisation and the correction of the plane/line vertex position by projection.

Evaluation of the algorithm with synthetic data showed us that finding the best possible fit to the data does not necessarily provides us with the best orientations of the planes. Instead, we can enforce orientations on the planes and keep a fairly good fit to

Model	Planes	Lines	Parameters	Constraints	Time (s)
Synthetic	3	10	45	6	1
Factory 1	3	21	83	6	12
Factory 2	3	54	205	6	21
Arenberg castle	5	0	15	7	1
Farmhouse	6	63	295	13	31
Mosque Inside	22	112	418	234	172
Mosque Outside	33	87	374	277	97

Table 7.3: The table summarises the optimised models in this chapter. It shows the model name, the extracted planes/lines, the size of the model description (parameter vector), the applied constraints and the execution time of the optimisation algorithm (from left to right).

the data.

The plane and line extraction works fairly robustly for all models. One has to specify a distance tolerance for each of the extractions. However, analysing the extracted planes to ensure connectivity was not computationally feasible with very large datasets such as the mosque data. For the line extraction it might be necessary to limit the initial edge set to edges at plane borders if many detailed structures or noisy surfaces are present in the scene. Otherwise, lines are extracted that go straight through detailed structures. We demonstrated the plane labelling on the castle and the farmhouse scenes. One unique labelling exists for the planes in these scenes. However, no labelling was possible for the factory scenes, because the planes did not form a shape of a house. No unique labelling for the mosque was possible, either, because it is not obvious which planes are part of the walls, ceiling or floor. For the other architectural scenes, the peaked roof restricted the labelling to one unique solution.

We used several scenes such as the factory scenes, the castle and the farmhouse that are simple and very structured. The mosque scenes were far more complex. The optimisation worked fine for the simpler models that include only a few planes and a few tens of lines. However, the complex mosque scenes show some limitations of our

optimisation algorithm. The dimension of the search space (number of variables) is not a problem. Instead, the number of constraints used (231 and 274 for the indoor and outdoor mosque models) makes the optimisation difficult for the Downhill Simplex algorithm. The model is over constrained. The use of so many constraints deforms the search space in a way that makes it difficult for the Downhill Simplex to reach a place close to the global optimum. One can overcome this problem by using less constraints or another optimisation technique such as a truly global GA which can handle over-constrained models better. Another way to overcome the problem is to limit the number of constraints by clustering parallel planes and assigning orthogonality constraints between the clusters.

Our last scene included some walls that are aligned at 45 degrees to each other. This special case occurs at the corners of the mosque. In general vertices can belong to different planes. If they lie on plane borders, they are projected twice. The double projection of the vertices on the 45 degree plane borders interferes to some degree with the plane that is flattened first. These interferences can be avoided by projecting the vertices onto the intersection of the planes.

Chapter 8

Conclusion

Reconstructing accurate 3D models of existing objects has been a long term goal in the field of computer vision. In particular, the reconstruction of architecture has proved to be very popular [DTM96, Pol99] because of its common appearance. However, architecture is a difficult object to reconstruct, because of its size, frequent occlusions and varied lighting conditions when reconstructed from photographs. These difficulties lead to irregularities in the created 3D models.

8.1 Summary and discussion

The research presented in this thesis has focused on the application of geometrical knowledge in form of architectural constraints to improve architectural 3D mesh data by removing irregularities in the 3D models. Thus, accuracy of the models is improved. We analysed where and which kind of constraints have been used in the literature. Previous work has used architectural constraints in 3D reconstruction for camera calibration [SM99, WBS01] and reconstruction from single [LCZ99, CRZ00, WBS01] and multiple [FRL+98, BZ00, DTRC01, WZ02b] photographs. For single image reconstruction, it is essential to know prior geometrical information about the scene. The novelty of the work presented here is to apply architectural constraints to architectral 3D data instead of applying them to photographs. We can use constraints more comprehensively, because our models are not limited by image based reconstruction

methods. Our techniques are related to those used for constrained reconstruction of industrial parts as presented in [WFAR99b, RFWA00, MLMM01, LMM02].

We investigated architectural principles to derive a set of constraints likely to be present in architecture. These constraints are used to make architectural features more regular in terms of their architectural properties. We selected parallelism and orthogonality constraints on planar regions (typically walls, floors and ceilings in architecture) and lines (linear structures such as columns, outline of windows, intersections of walls). Furthermore, linear structures in architecture often form a three-dimensional structural frame that is characterised by the three principal directions. Orthogonal and parallel structures are found in many architectural works of many different kinds and characters. The constraints enforce the proper orthogonal or parallel alignment of planes and lines. Furthermore, triangulated regions that are almost planar are made coplanar and lines that are almost straight are made straight.

We applied the constraints to a raw triangulated mesh. Zabrodsky concluded in [ZW97] that corrections following meshing generally give a greater improvement. Our techniques rely entirely on the 3D information present in the meshes. The use of triangular meshes allows us to exploit mesh properties such as the edge length and the triangle size for feature extraction.

Our approach is a three stage process. First, architectural features are extracted from previously triangulated 3D models. A robust probabilistic algorithm (RANSAC) was used to extract planar patches and lines from triangular meshes. The extraction depends on distance thresholds to specify how much 3D structure lies on a plane or line. We have shown that conservatively loose distance thresholds lead to a robust extraction. However, correct extraction at feature intersections can be difficult depending on how well-distinguished the features are and how much noise is on them. The problem at the feature extraction stage is to distinguish between noisy data and small features in our data. All vertices that lie within the distance threshold of a feature collapse onto the feature at the optimisation stage. Any structure smaller than the distance threshold will therefore disappear from the 3D data. However, since the 3D model normally also contains intensity texture (in which, of course, the structure remains visible) the visual appearance of the model is not necessarily compromised. The parameters of all

extracted features are concatenated in a parameter vector, that is the model description.

The second stage is the automatic discovery of the constraints. Little work has been done on the automatic discovery of the constraints in the past leaving it often to the user to specify them. We label the planes of an architectural model by matching them with a tree-search strategy against a structural semantic net of a generic house. The planes are labelled as wall, floor, ceiling and roof. Suitable constraints are selected to enforce certain relationships between planes. A wall is orthogonal to the ground plane for example. However, not all architectural models have the shape specified in the semantic net. The labelling of such scenes is ambiguous. The extracted lines are grouped together by complete linkage clustering. The three principal directions of the architectural scene are extracted by cluster analyses of the lines. We assign 3 constraints to enforce the orientation of the three directions.

The model description provided by the feature extraction is optimised taking the selected constraints on the planes and lines into account. We defined an evaluation function leading to a model that satisfies all constraints and resembles the original model closely. We use the Downhill Simplex algorithm, which is a robust numerical optimisation algorithm. It has been shown that the Downhill Simplex optimisation is computationally faster and more precise at the final exploration than the GA. The Downhill Simplex has been shown to converge to a point at least close to the global optimum. However, it is essential that we start the Downhill Simplex with a good initial estimation of the model description, which is provided by the feature extraction in our case.

We have evaluated our methodology on models created from both synthetic and real scenes. Experiments have been performed with synthetic data that take 3D noise on the features into account. The evaluation of these experiments has enabled us to conclude that we can constrain planes and lines in the model and retain a good fit to the original data. Several 3D models reconstructed from real data with differing complexity have been used. We demonstrated our three stage process on these models. Our process works well for the factory and simple house models, but shows its limitations in the much more complex mosque scenes. The mosque models are much larger in terms of both numbers of triangles and numbers of structural elements such as walls. The large

number of constraints between the large number of planes in particular led to a less optimal constraint satisfaction.

Two publications have resulted directly from the work on constrained architectural reconstruction:

- H. Cantzler and R.B. Fisher and M. Devy. Quality enhancement of reconstructed 3D models using coplanarity and constraints. Symp. for Pattern Recognition (DAGM), Zurich, Switzerland, 34-41, 2002
- H. Cantzler and R.B. Fisher and M. Devy. Improving architectural 3D reconstruction by plane and edge constraining. British Machine Vision Conference (BMVC), Cardiff, UK, 43-52, 2002

8.2 Contributions

The main original contributions of this thesis can be outlined as follows:

- A method for the identification and optimisation of the three principal directions that form the three-dimensional structural frame of the architecture (simultaneously published with [WZ02b]).
- A method for the semantic labelling of large planar surfaces such as walls, floors, ceilings and roofs in architectural scenes. We identify them by matching a semantic net of generic building features and their relations against the features using a tree-search strategy.
- A method for the automatic discovery of architectural constraints. Often the user has to specify them, but we showed how an automatic strategy can be used to discover constraints between features.
- A method for the constrained repositioning of architectural features such as walls, floors and linear elements to conform to model data and architectural constraints.

- The final contribution of this research is a methodology for application of constraints to architectural models. Before, there had been limited work in the area of constrained reconstruction of architecture from 3D data. We demonstrated how our approach removes imperfections such as small irregularities on planes and lines. Furthermore, the orientations of planes and lines are corrected. As a result the model accuracy is improved and the visual appearance is enhanced.

8.3 Future research

Despite the advances made by the research presented in this thesis towards constrained reconstruction, this area of research is by no mean finished. Further advances could be made in the areas of feature extraction and model interpretation to enable us to identify, for example, windows, doors and columns. One could extend our approach to constrain window, door and columns with shapes templates. Windows and doors are rectangular and columns have a cylindrical shape. These features are all parallel to the wall and orthogonal to the base plane. Furthermore as windows often have the same depth we could identify the window plane.

The extraction of features and the labelling of planes could be improved by taking texture information into account. Texture can guide plane and line extraction where evidence of planes and lines is weak from the 3D data. Texture information could also be used to assist the labelling of the house planes. In a reconstructed architectural environment one could exploit, for example, the property that house roofs or walls have a certain texture.

Perhaps the most important limitation of the developed algorithm is its difficulty with dealing with more complex models. The computational speed of the connectivity checking during the plane extraction depends heavily on the number of the triangles present in the planar patch. For large models such as the mosque models, one needs to find another way to check connectivity for a realistic computational speed. Furthermore, the mosque models with their many planes are heavily over-constrained. We can solve this problem by clustering the planes in a similar manner to the line clustering. Parallel planes are implicitly parallel and the size of the parameter vector would be

smaller, since parallel planes share the same orientation after clustering. More importantly, the number of constraints for scenes with many planes would be much smaller.

Also, the outdoor mosque model contains walls that intersect at 45 degrees. This special case occurs at the 4 corners of the mosque. Projection of the points at the wall intersections led to small defects on some of the surfaces. To improve point projection, one could extract the points that lie on the intersections and project them onto the optimised intersections instead of simply projecting them onto the separate planes.

Appendix A

Random Sample Consensus (RANSAC)

The RANSAC algorithm is an algorithm for robust fitting of models. It was introduced by Fischler and Bolles in 1981 [FB81]. It is robust in the sense of good tolerance to outliers in the experimental data. It is capable of interpreting and smoothing data containing a significant percentage of gross errors. The estimate is only correct with a certain probability, since RANSAC is a randomised estimator. The algorithm has been applied to a wide range of model parameters estimation problems in computer vision, such as feature matching, registration or detection of geometric primitives.

A.1 Subsampling of the input data

The structure of the RANSAC algorithm is simple but powerful. First, samples are drawn uniformly and at random from the input data set. Each point has the same probability of selection (uniform point sampling). For each sample a model hypothesis is constructed by computing the model parameters using the sample data. The size of the sample depends on the model one wants to find. Typically, it is the smallest size sufficient to determine the model parameters. For example, to find circles in the data set, one has to draw three points, since three points are required to determine the parameters of a circle. Drawing more than the minimal number of sample points is

inefficient, since the probability of selecting a sample consisting only of inlying data points (*i.e.* all data points belonging to the same model), that gives a good estimate and at random, decreases with respect to the increasing sample size. Thus the minimal sample set maximises the probability of selecting a set of inliers from which later a good estimate will be computed.

A.2 Hypotheses evaluation

In the next step, the quality of the hypothetical models is evaluated on the full data set. A cost function computes the quality of the model. A common function is to count the number of inliers (*i.e.* data points which agree with the model within an error tolerance). The hypothesis which gets the most support from the data set gives the best estimate. Typically, the model parameters estimated by RANSAC are not very precise. Therefore, the estimated model parameters are recomputed by for example a least-squares fit to the data subset which supports the best estimate. The input data may support several distinct models. In this case, the model parameters for the first model are estimated, the data points supporting the model are removed from the input data and the algorithm is simply repeated with the remainder of the data set to find the next best model. The strength of the algorithm is that it is likely to draw at least one set of points which consists only of inliers and thus results in a good estimate of the model parameters.

A.3 Process variables

The RANSAC technique uses three variables to control the model estimation process. The first determines whether or not a data point agrees with a model. Typically, this is some error tolerance that determines a volume within which all compatible points must fall in. The number of model hypotheses that are generated is the second variable. It depends on the probability to draw a sample including only inlying data points. As the proportion of outliers and the minimal sample set size increase the number of model hypotheses must be increased to obtain a good estimate of the model parameters.

The proportion of outliers depends on the noise level and on how many models are supported by the data set. Furthermore, one tolerance variable is needed to determine if a correct model has been found. An extracted model is deemed valid if there is sufficient support from the data points for this model. A valid circle has been found in the data if for example at least 20 data points are found which lie close enough to the circle. In case of multiple models in the data, more models are extracted until there is insufficient support for any more models.

A.4 Runtime improvements

The computational efficiency of the algorithm can be improved significantly in several ways. The speed depends on two factors: firstly, the number of samples which have to be drawn to guarantee a certain confidence to obtain a good estimate; and secondly, the time spent evaluating the quality of each hypothetical model. The latter is proportional to the size of the data set.

Typically, a very large number of hypotheses are created from contaminated samples (*i.e.* samples containing outliers). Such models are consistent with only a small fraction of the data. The evaluation of the models can be computationally optimised by randomising the evaluation [CM02]. Every hypothetical model is first tested only with a small number of random data points from the data set. If a model does not get enough support from this random point set, then one can assume with a high confidence that the model is not a good estimate. Models passing the randomised evaluation are then evaluated on the full data set.

The performance of the algorithm degrades with increasing sample size or in case multiple models are supported by the data due to the decreasing probability of sampling a set that is composed entirely of inliers. A common observation is that outliers possess a diffuse distribution. In contrast, inliers will tend to be located closely together. Therefore, the uniform sampling of points is replaced by selection of sample sets based on proximity taking spatial relationships into account [MTN+02]. The first initial sample point is selected randomly. The rest of the points are random points lying within a hypersphere centred on the first point. The selection of sample sets of adjacent

points can significantly improve the probability of selecting a set of inlying points and thus drastically reduce the number of samples required to find a good model estimate.

Appendix B

Clustering

Organising observed multi-dimensional data into meaningful structures is a common task which is vital in many scientific and commercial fields. Cluster techniques are utilised to divide a large set of objects into separate classes (also called clusters, groups or partitions) of densely populated regions in the data space.

The data space is usually not uniformly occupied. Generally, clustering techniques are to be seen as tools for the exploration of the data space. They identify the sparse and the crowded places and hence discover the overall pattern of the dataset distribution. The dataset is split according to some object variables, which are frequently the result of measurements.

The two major types of classification techniques are non-hierarchical (partitioning) and hierarchical clustering (tree clustering) [JD88]. The best known partitioning technique is k-means partitioning. K-means partitioning is based on initially specifying a number of classes. Each class has a seed point and all objects within a prescribed distance are included in that class. Objects are moved between those classes with the goal of minimising variability within classes and maximising variability between classes. The best-known criterion is to minimise the sum of the squared distances between all elements of a class.

Hierarchical clustering results in hierarchies of nested partitions. The clusters themselves are repeatedly grouped to larger clusters. In contrast to the non hierarchical techniques, these clusters are not defined a priori but are created by the clustering algorithm. Data spaces from different clustering problems may have different mathematical

properties that influence which clustering strategy may be applied. Therefore, the distance function (similarity between two objects in the dataset) and the chosen clustering strategy greatly determine the resulting clustering. We focus in this appendix on the hierarchical clustering techniques as they were used in this research.

B.1 Hierarchical clustering

Hierarchical cluster techniques are used when a stratified structure of clusters at different heterogeneity levels are desired. Divisive clustering (top-down) starts from the entire data set and iteratively splits it until every class consists of one object only. Agglomerative clustering (bottom-up) goes the other way around, as follows. First, each object represents its own cluster. Then, the distance function is used to find the pair of clusters which is most similar (closest distance to each other). This pair is merged to form a new bigger cluster. So, clusters are grouped together to form larger and larger clusters. As clusters get larger and larger more distant clusters are linked together and their elements become increasingly dissimilar. At every stage one wants the two most similar clusters to be merged. The algorithm terminates when all objects are combined to one cluster (the entire data set) or the degree of dissimilarity reaches a certain threshold.

B.2 Distance measurements

As a fundamental requirement, a notion of distance has to be introduced in the object space. This means we need to define the similarity or distance function between the individual objects of a data set. This distance depends on the mathematical properties of the data space. It can be based on a single dimension or multiple dimensions. The most common way of computing distances between the two objects o_1 and o_2 in a multi-dimensional space is to compute the Euclidean distance $\sqrt{(o_1 - o_2)^2}$. However, often other derived measures of distance are more suitable for applications.

The most commonly used Euclidean distance (as used in chapter 4) can be squared to get the Squared Euclidean distance. This distance places progressively greater

weight on objects that are further apart. Many other distance measurements such as the Manhattan (City-block) and Chebychev distance can be used as a similarity measurement. Usually, all the distances between the objects are calculated once and are then saved in a distance matrix.

B.3 Linkage algorithms

The measurements in the distance matrix are distances between single objects in our data set. However, once several objects have been grouped together in clusters, how do we determine the distance between those clusters? We need to define the distance between clusters as well. There are various possible linkage algorithms which differ in how they derive cluster distances from the distances of the objects.

Single linkage clustering defines the distance between two clusters as the minimal distance of any two objects belonging to different clusters (nearest neighbour method). This method is used in chapter 4 to check extracted planes for connectivity. Single linkage clustering is best suited to detect chains or elongated structures. However, it is less suitable for isolating spherical or poorly separated clusters. Complete linkage clustering is opposite to single linkage clustering as it uses the maximal distance of objects in different clusters (furthest neighbours). We use this linkage algorithm in chapter 5 to group parallel lines together. All entries in a cluster are linked to one another within some minimum similarity. This method usually performs well in cases when the objects actually form naturally distinct clumps in the data space. The resulting clusters have spherical shapes, where all members of a class are tightly bound together. This method is inappropriate if the data tend to form rather elongated clusters. In between the two previous methods is the average linkage clustering. It uses the average distance of the pairwise links between the two clusters based on all objects in the clusters. The resulting clusters are intermediate in tightness of single linkage and complete linkage. Many other methods are possible. Some use the centroids or medians to decide which clusters to merge.

Appendix C

Publications

- H. Cantzler and R.B. Fisher and M. Devy. Quality enhancement of reconstructed 3D models using coplanarity and constraints. Symp. for Pattern Recognition (DAGM), Zurich, Switzerland, 34-41, 2002
- H. Cantzler and R.B. Fisher and M. Devy. Improving architectural 3D reconstruction by plane and edge constraining. British Machine Vision Conference (BMVC), Cardiff, UK, 43-52, 2002

Quality enhancement of reconstructed 3D models using coplanarity and constraints

H. Cantzler[1], R.B. Fisher[1] and M. Devy[2]

[1] Division of Informatics, University of Edinburgh,
Edinburgh, EH1 2QL, UK
{helmutc,rbf}@dai.ed.ac.uk
[2] LAAS-CNRS, 31077 Toulouse, France
michel@laas.fr

Abstract

We present a process to improve the structural quality of automatically acquired architectural 3D models. Common architectural features like orientations of walls are exploited. The location of these features is extracted by using a probabilistic technique (RANSAC). The relationships among the features are automatically obtained by labelling them using a semantic net of an architectural scene. An evolutionary algorithm is used to optimise the orientations of the planes. Small irregularities in the planes are removed by projecting the triangulation vertices onto the planes. Planes in the resulting model are aligned to each other. The technique produces models with improved appearance. It is validated on synthetic and real data.

Keywords

Surface geometry, Shape, Scene analysis, Constrained architectural reconstruction

1 Introduction

The process of 3D reconstruction is often affected by noise in the measurements. Furthermore, inaccuracies are created by view merging, segmentation and surface fitting. One way to improve the reconstruction is to use more sophisticated methods like photogrammetry techniques. Another way is to exploit properties of the scene. Architectural scenes are particularly suitable for the application of constraints since the geometry is typically very structured. Architectural constraints can be used for 3D reconstruction from single [15, 7] or multiple [4, 1] intensity images. Features used for architectural constraints are typically straight lines, large coplanar regions and the parallelism and orthogonality of lines or planes. These kinds of features can be easily found in architecture scenes. In [3] research is described that improves architectural 3D models by automatically straightening edges. The work presented in this paper concentrates on extracting planar regions and applying coplanar, parallelism and orthogonality constraints more comprehensive then in previous work to the full 3D model. We apply the constraints to the data following meshing. Zabrodsky concluded in [16] that corrections

following meshing generally give a greater improvement. Our method is independent of the calculation of the 3D structure unlike the work presented in [15, 7, 4, 1] where constraints are used in combination with reconstruction from intensity images.

This work consists of three steps. First, architectural features are extracted from already triangulated 3D models (Section 2). We use a RANSAC technique [5] to find planes in the model (similar to [2]). The next step is the automatic extraction of the constraints out of the scene. Few papers have dealt with the automatic extraction leaving it to the user to specify them [11, 14]. The interpretation of the scene is formalised as a constraint satisfaction problem [13]. Liedtke used a semantic net for interpretation of architectural scenes [8]. His interpretation is hypothesis driven. Hypotheses are verified or falsified by matching the 3D objects against the image. In our work we match the planes against a semantic net of a house by using a backtracking tree search (Section 3). The semantic net concentrates on the definition of the 3D objects and its relations. We check the interpretations only by verifying the relationships between the 3D objects. Constraints are assigned to almost-regularities like parallel or orthogonal walls. The last and final step consists of applying the constraints to the model (Section 4). The original model is fitted to the new constrained model. Optimising the model can be done in a number of ways (*e.g.* numerically [2, 14] or evolutionary [11]). We use an evolutionary approach. The model and the constraints are passed to the GenoCop 5 algorithm, proposed by Michalewicz [9]. The vertices are projected onto the planes after finding the optimal parameters. The result is a model with fewer irregularities (*e.g.* edges on walls) and aligned walls.

2 Feature detection

At all stages of the process, the model is a mesh consisting of vertices $V = \{(x, y, z)'\}$ and triangles $T = \{(v_1, v_2, v_3)\}$. The first step is to extract planes from the raw triangulated model. Before starting the extraction the model is normalised. It is mapped into an unit sphere at the origin. A robust RANSAC algorithm [5] is then used to obtain a set of planes. The algorithm generates a number of random plane hypothesis from the points in V. The distance of a triangle centroid to the hypothetical plane is calculated by computing the difference between the distance of the plane to the origin D and the dot product between the triangle centroid $C = (c_x, c_y, c_z)'$ and the unit plane normal $N = (n_x, n_y, n_z)'$. Triangles that satisfy the following inequality belong to the hypothetical plane.

$$|C \cdot N - D| < tolerance \tag{1}$$

The size of a hypothetical plane is calculated by adding up its triangle sizes. The hypothesis that creates the largest plane is selected. The exact number of planes in a model is not known. So, we repeat the RANSAC algorithm until the size of the resulting plane falls under a certain threshold. (An EM algorithm could instead have been used to select the number of planes and fit them, but we chose a simpler technique to focus on the reconstruction issues.)

This technique gives reasonable results. However, it sometimes produces a plane that consists of small disconnected patches distributed over the scene. An architectural

plane (*e.g.* a wall) is not usually separated by a large gap. However small gaps frequently occur for example due to the presence of pipes or decorations. Therefore, the planes are analysed by single linkage clustering [6] to ensure that the triangles of a plane are closely connected. The cluster technique starts with the individual triangles and groups them together to form larger and larger clusters (hierarchical clustering). The distance between two clusters is defined as the minimal Euclidean distance of any two triangles belonging to different clusters (nearest neighbor method). The clustering terminates after reaching a certain distance. This distance specifies how far apart parts of the plane can be.

3 Scene interpretation

We interpret the scene using the features (planes) found previously. A model of an architectural scene is described in a semantic net (see figure 1). The model entities are represented as nodes in the net. The nodes are connected via different types of relationships. A semantically meaningful description is assigned to the scene features by matching them to the semantic net. A backtracking tree search is used to find the best match. The algorithm takes as input a set of features F, a set of possible model labels L and a set of binary model relationships R which limits the possible labelling. The tree search starts with the first feature from F and assigns all labels from L. A second feature is fetched from F and all labels are assigned. At this level some of the labels might be ruled out because they violate the given relationships. This process continues until all features have been labelled. A consistent labelling then exists if each feature is assigned a valid label that is also arc consistent with adjacent nodes. The relationships between features are used to select appropriate geometrical constraints for enforcing parallelism or orthogonality later in the optimisation step.

The model-entities (labels) and the relationships among the entities represent the knowledge of a typical architectural scene. Possible labels are L = {*Side Wall, End Wall, Base Plane, Ceiling/Floor, Roof, No Feature*}. The binary relationship functions check if the architectural relationship between two features and their labels is valid (*e.g.* horizontal and vertical walls are almost perpendicular). Angle relationships between two features are checked with a certain tolerance (3 degrees). The ”Above” relationship is satisfied if 99% of the vertices of one plane are above a second plane defined by surface normal and distance. *No Feature* does not have any relation with a normal feature and can therefore be assigned everywhere. The final labelling is obtained by finding the solution that maximises the number of architectural labels.

The semantic net models a reasonable subset of all houses. It includes the interior and exterior structure of houses. The model can include an arbitrary number of walls. They can be on the same level or on different ones (then seperated by a *Floor/Ceiling*). The base plane is below all other parts of the building. It represents the ground on which the house stands. The roof is modelled as a typical sharp roof. Errors in the scene description are resolved by labelling them as *No Feature*. The semantic net can be easily extended with features like windows and doors. These features can be modelled as parallel and close to the actual walls. However, the previous plane detection concentrates on finding big planes. So, modelling windows and doors is not necessary at this step.

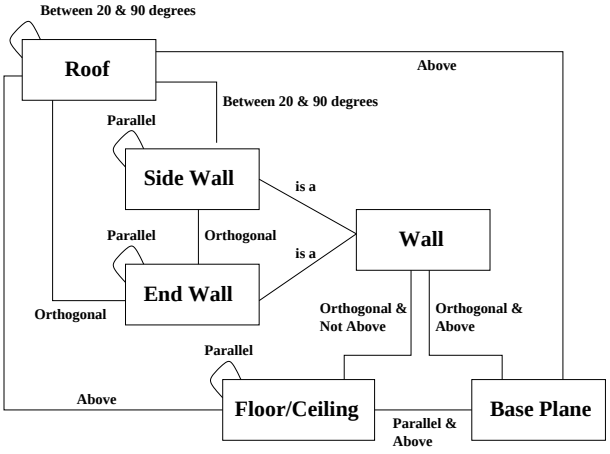

Fig. 1. The model of the architectural scene is represented by a semantic net. Nodes represent the model entities and are linked by architecturally meaningful relationships.

4 Model optimisation

Optimising the model by enforcing the constraints found previously is formulated as a nonlinear programming problem. There are many algorithms which are designed to search spaces for an optimum solution. Some of them become ill-conditioned and fail with nonlinear problems. We use the GenoCop 5 algorithm developed by Michalewicz [9]. It is a genetic algorithm (GA) which uses real-value genes and includes methods to deal with linear, non-linear, inequality and domain constraints.

The GA uses the parameter vector $\boldsymbol{p}$ which concatenates all the parameters for the individual planes as the chromosome. The evaluation function consists of the squared residuals of the vertices and the constraint functions. The squared residual is the squared geometric distance from the mesh vertices $\{x_{i,j}\}$ to their planes $\{P_i\}$. The residual of every plane is normalised with its number of vertices N_i. Thus, model size does not affect results. Every constraint is represented by a constraint function c(). The values of these functions correspond to the degree that the constraints are satisfied. The constraint functions can be seen as a penalty functions. λ is a weight factor which scales the constraints to the residuals.

$$\sum_i \frac{1}{N_i} \sum_j dist(P_i(\boldsymbol{p}), x_{i,j})^2 + \lambda \sum_i c^{(i)}(\boldsymbol{p}) \tag{2}$$

Additionally, constraints are used to narrow the search space of the evolutionary algorithm. Domain constraints are applied to individual components of the surface normals and the distances. Each of the parameters can never be outside the range [-1,+1]

since the 3D model is mapped into a normal sphere at the origin. Furthermore, unity constraints are applied to the surface normals N.

So far we have obtained the optimised model parameters. We now project the vertices of the planes onto their planes. We calculate the new coordinates $V_p = (x_p, y_p, z_p)'$ of the vertex with the original vertex $V = (x, y, z)'$, the unit surface normal of the plane $N = (n_x, n_y, n_z)'$ and the distance D of the plane to the origin as:

$$V_p = V - tN \tag{3}$$

where

$$t = \frac{V \cdot N - D}{N \cdot N} \tag{4}$$

5 Experimental results

The proposed technique described above is general. It is independent of the way the 3D model was created (*i.e.* from range or intensity data) and of model properties like variance of the triangle size. It has been applied to several triangulated models. We will here present results for a synthetic model and for two reconstructed real models.

First, we applied the described technique to the synthetic model. The model consists of a perfect mesh of three walls at 90 degrees (1323 vertices & 2400 triangles). Two walls are parallel. A varying amount of Gaussian distributed 3D noise is added to the vertices. The first graph shows the angle error from plane extraction (top curve), improving the plane fit (middle curve) and application of constraints (bottom curve, near noise level axis). Improving the plane fit is done without using any constraints in the evaluation function. The angle error from plane extraction is a result of the random nature of RANSAC. Improving the fit using all data points from the planes gives much better results. Finally, using the constraints gives an angle error very close to zero. The second graph shows the mean squared residual after plane extraction (top curve), improving the fit (dashed curve) and constraining the model (solid curve). The parameters obtained from RANSAC show the biggest error. The mean residuals from improving the fit and from applying the constraints are fairly similar and are both significantly below the the RANSAC curve. The two graphs show that applying constraints improves the orientation of the walls without worsening the fit.

We show an experiment with the reconstructed model of Arenberg castle (in Belgium) reconstructed by the Catholic University of Leuven [10]. The model was reconstructed from an image sequence of 20 images (6292 vertices & 12263 triangles). The walls and the ground on the original solid model show clearly a lot of small irregularities (see figure 4). 5 planes are extracted (3 walls, 1 floor and 1 roof). The planes are constrained by 7 constraints. The angles between the planes vary from the optimum by 1.5 degrees on average before optimisation. After optimisation they differ less than 0.01 degrees. The result shows the model with removed irregularities and constrained planes. The average disparity of the moved vertices as a fraction of the model diameter is 0.33%. The optimisation step took 54 seconds on an Intel Celeron with 400MHz.

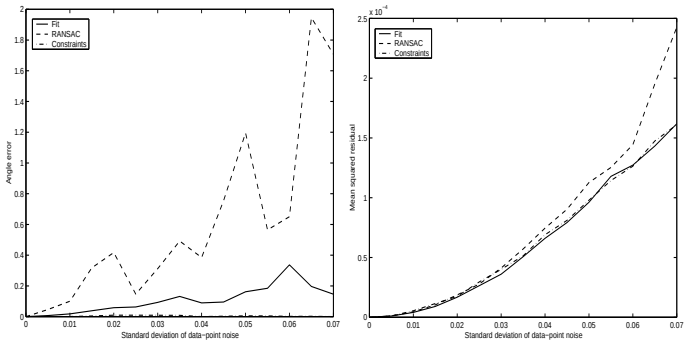

Fig. 2. Results for the synthetic model. The left graph shows the angle error in degrees versus the noise level. The graph on the right shows the mean squared residual versus the noise.

Next, we briefly describe results for a Bavarian farmhouse reconstructed by the European Commission Joint Research Centre (JRC) [12]. It was reconstructed from multiple range data scans (12504 vertices & 16589 triangles). This is a full 3D model. The plane extraction finds 4 walls and two planes for the roof. The orientations of the walls are already fairly good. The angles between the planes differ on average by 0.5 degrees in the original model. After optimisation they differ less than 0.01 degrees. The original solid model shows small edges on the walls. The result has these edges projected onto the wall (see figure 3 for a close view of a wall).

Fig. 3. A close view of a wall of the farmhouse. On the left is the unconstrained model. Surface ripples are most easily seen in the circled areas. On the right is the optimised model.

6 Conclusion and future work

Previous work used architectural constraints mainly for scene reconstruction from intensity images. This work shows how architectural constraints can be used for improving the reconstruction of full 3D models independent of the sensor data. Only 3D information is used. The constraints make architectural features more regular in terms of their architectural properties. We exploit common architectural features like walls and their relationships to each other.

Initially, a RANSAC technique obtains a set of planes from the 3D data. We automatically discover the graph of constraints between the planes by using a tree search strategy. Even conservatively loose thresholds on angles and position lead to a correct labelling of the planes in the scene. The model parameters are optimised with a robust evolutionary algorithm. A numerical normalisation of the model beforehand leads to domain constraints on the parameters which speeds up the search algorithm. The experimental results show how imperfections like small irregularities on planes and the orientations of walls are corrected. The visual appearance of the model is enhanced.

Future work aims at incorporating edges into the process of model optimisation. This includes extraction of edges in the model, straightening of edges and the use of parallelism or orthogonality constraints where applicable.

References

1. C. Baillard and A. Zisserman. A plane-sweep strategy for the 3d reconstruction of buildings from multiple images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 33(B2):56–62, 2000.
2. A. Bartoli. Piecewise planar segmentation for automatic scene modeling. *Conference on Computer Vision and Pattern Recognition, Hawaii*, pages 283–289, 2001.
3. P. Dias, V. Sequeira, J.G.M. Goncalves, and F. Vaz. Combining intensity and range images for 3d architectural modelling. *International Symposium on Virtual and Augmented Architecture, Dublin*, pages 139–145, 2001.
4. A. Dick, P. Torr, and R. Cipolla. Automatic 3d modelling of architecture. *British Machine Vision Conference, Bristol*, pages 372–381, 2000.
5. M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
6. A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.
7. D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. *Eurographics*, 18(3):39–50, 1999.
8. C.-E. Liedtke, O. Grau, and S. Growe. Use of explicit knowledge for the reconstruction of 3d object geometry. *International Conference on Computer Analysis of Images and Patterns, Prague*, pages 580–587, 1995.
9. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1996.
10. M. Pollefeys. *Self-calibration and metric 3D reconstruction from uncalibrated image sequences*. PhD thesis, University of Loewen, 1999.
11. C. Robertson, R.B. Fisher, N. Werghi, and A. Ashbrook. Fitting of constrained feature models to poor 3d data. *Proceedings Adaptive Computing in Design and Manufacture, Plymouth*, pages 149–160, 2000.

12. V. Sequeira, K. Ng, E. Wolfart, J.G.M. Goncalves, and D.C. Hogg. Automated reconstruction of 3d models from real environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54:1–22, 1999.
13. D.L. Waltz. *Generating Semantic Descriptions from Drawings of Scenes with Shadows*. PhD thesis, AI Lab, MIT, 1972.
14. N. Werghi, R.B. Fisher, A. Ashbrook, and C. Robertson. Shape reconstruction incorporating multiple non-linear geometric constraints. *Computer-Aided Design*, 31(6):363–399, 1999.
15. M. Wilczkowiak, E. Boyer, and P.F. Sturm. Camera calibration and 3d reconstruction from single images using parallelepipeds. *International Conference on Computer Vision, Vancouver*, pages 142–148, 2001.
16. H. Zabrodsky and D. Weinshall. Using bilateral symmetry to improve 3d reconstruction from image sequences. *Computer Vision and Image Understanding*, 67(1):48–57, 1997.

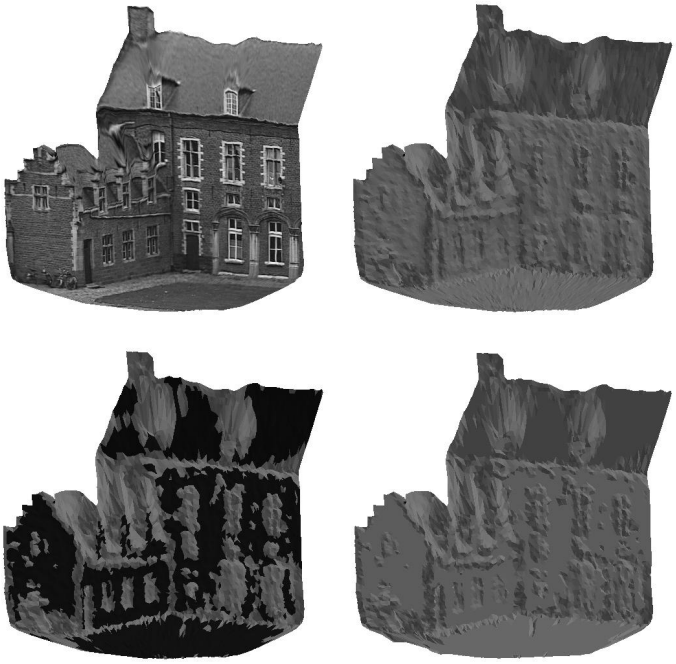

Fig. 4. The textured model (top/left), the original solid model (top/right), the extracted planes (bottom/left) and the resulting model after optimisation (bottom/right) from the castle. The extracted planes are displayed a bit darker than in the solid model.

Improving architectural 3D reconstruction by plane and edge constraining

H. Cantzler[1], R.B. Fisher[1] and M. Devy[2]

[1] Division of Informatics, University of Edinburgh,
Edinburgh, EH1 2QL, UK
[2] LAAS-CNRS, 31077 Toulouse, France

{helmutc,rbf}@dai.ed.ac.uk, michel@laas.fr

Abstract

This paper presents new techniques for improving the structural quality of automatically acquired architectural 3D models. Common architectural features like parallelism and orthogonality of walls and edges are exploited. The location of these features is extracted from the model by using a probabilistic technique (RANSAC). The relationships among the planes and edges are inferred automatically using a knowledge-based architectural model. A numerical algorithm is used to optimise the orientations of the features. Small irregularities in the model are removed by projecting the triangulation vertices onto the features. Planes and edges in the resulting model are aligned to each other. The techniques produce models with improved appearance. We show results for synthetic and real data with consideration of noise.

Keywords

Surface geometry, Shape, Scene analysis, Constrained architectural reconstruction

1 Introduction

The process of 3D reconstruction of scenes is often affected by noise in the measurements. Furthermore, inaccuracies are created by view merging, segmentation and surface fitting. Ways to improve the reconstruction are to use more sophisticated methods like photogrammetry techniques or to increase the number of views (possibly from different sensors). Another way is to identify and exploit properties of the scene to improve the model. 3D reconstruction of industrial parts has used prior knowledge to improve models for a long time [14, 17]. Architectural scenes are also particularly suitable for the constraints since the geometry is typically very structured [3]. Architectural constraints can be used for camera calibration as well as 3D reconstruction from single [18, 10] and multiple [6, 1] intensity images or for image based modelling of buildings [4]. Features used for architectural constraints are typically straight lines, large coplanar regions and the parallelism and orthogonality of lines or planes. These kinds of features can be easily found

in architecture scenes. Architectural constraints for improving 3D models have been used to automatically straighten edges in 3D models [5]. The work presented in this paper combines parallelism and orthogonality constraints on planar regions (typically walls, floors and ceilings) and edges. Planes are flattened and edges are straightened. We enforce the alignment of planes and edges by applying orientation constraints. Triangulated nearly planar regions are made coplanar and edges straight. We apply the constraints to the data following meshing. Zabrodsky concluded in [19] that corrections following meshing generally give a greater improvement. The method is independent of the calculation of the 3D structure unlike the work presented in [18, 10, 6, 1] where constraints are used in combination with reconstruction from intensity images.

This work consists of three steps. First, architectural features are extracted from previously triangulated 3D models (Section 2). We use RANSAC techniques [7] to find planes (similar to that given in [2]) and edges in the model. The next step is the automatic extraction of the relationships between the extracted scene features. Few papers have dealt with the automatic extraction leaving it to the user to specify them [14, 17]. The interpretation of the scene is formalised as a constraint satisfaction problem [16]. Liedtke used a semantic net for interpretation of architectural scenes [11]. His interpretation is hypothesis driven. Hypotheses are verified or falsified by matching the 3D objects against the image. In our work we match the planes against a semantic net of a generic house by using a backtracking tree search (Section 3). The semantic net concentrates on the definition of the 3D objects and its relations. We check the interpretations only by verifying the relationships between the 3D objects. We then analyse the edges in the models by clustering. Edges with similar orientation are clustered together. We obtain the three principle orientations of the architectural model from the clusters. Constraints are assigned to almost-regularities like parallel or perpendicular walls or edges. The last and final step consists of applying the architectural constraints (Section 4). The original model is fitted to the new constrained model. Optimising the model can be done in a number of ways (*e.g.* numerically [2, 17] or evolutionary [14]). We use the Downhill Simplex method [12]. It is a robust numerical multidimensional minimisation technique. After finding the parameter vector for the optimised model the vertices are projected onto the features. The result is a model with fewer irregularities (*e.g.* edges on walls) and aligned features (*e.g.* parallel walls).

2 Feature extraction

At all stages of the process, the model is a mesh consisting of vertices $V = \{(x, y, z)'\}$, edges $E = \{(v_1, v_2)\}$ and triangles $T = \{(v_1, v_2, v_3)\}$. The first step of the process is to extract planes and edges from the raw triangulated model. Before starting the extraction the model is normalised. The model is scaled and translated to fit into a unit sphere at the origin. A robust RANSAC algorithm [7] is then used to obtain a set of planes and edges.

2.1 Plane extraction

The algorithm hypothesizes a number of random planes from triples of points in V. The distance of a triangle centroid to the hypothetical plane is calculated by computing the difference between the distance of the plane to the origin D and the dot product between

the triangle centroid $C = (c_x, c_y, c_z)'$ and the unit plane normal $N = (n_x, n_y, n_z)'$. Triangles that satisfy the following inequality belong to the hypothetical plane.

$$|C \cdot N - D| < tolerance \tag{1}$$

The size of a hypothetical plane is calculated by adding up the areas of the triangles that satisfy (1). The hypothesis that creates the largest plane is selected. Each plane is represented by its minimal description, which is the surface normal represented by azimuth and elevation angles with respect to a reference vector and the distance to the origin. The exact number of features in a scene is not known. So, we repeat the RANSAC algorithm until the size of the resulting feature falls under a certain threshold. (An EM algorithm [13] could instead have been used to select the number of planes and fit them, but we chose a simpler technique to focus on the reconstruction issues.)

The plane extraction gives reasonable results. However, it sometimes produces a plane that consists of small disconnected patches distributed over the scene. An architectural plane (*e.g.* a wall) is not usually separated by a large gap. However small gaps frequently occur for example due to the presence of pipes or decorations. Therefore, the planes are analysed by single linkage clustering [9] to ensure that the triangles of a plane are closely connected. The cluster technique starts with the individual triangles and groups them together to form larger and larger clusters (hierarchical clustering). The distance between two clusters is defined as the minimal Euclidean distance of any two triangles belonging to different clusters (nearest neighbor method). The clustering terminates after reaching a certain distance. This distance specifies how far apart parts of the plane can be.

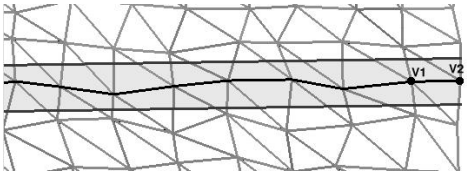

Figure 1: A fold edge goes horizontally thought this mesh. A hypothetical edge is created with the vertices V_1 and V_2. All edges that lie in the 3D corridor (grey area) belong to the hypothetical edge (middle dark line).

2.2 Edge extraction

The edge extraction starts with filtering the edges of the 3D model. Only jump edges (belonging to a single triangle) or fold edges, which separate non-coplanar triangles, are used. The algorithm creates random edges from the filtered edge set. The distance of a vertex to the hypothetical edge is calculated with the starting point $S = (s_x, s_y, s_z)'$ and the unit orientation $O = (o_x, o_y, o_z)'$ of the edge. All triangle edges with their vertices (V_1, V_2) that satisfy the following inequality belong to the hypothetical edge (see figure 1).

$$||(V_i - S) - O((V_i - S) \cdot O)|| < tolerance \tag{2}$$

The length of a hypothetical edge is calculated by adding up the lengths of the matched triangle edges. The hypothesis that creates the longest edge is selected. We represent every edge by its starting point S and orientations as α and β. We repeat the process until the length of the resulting edges falls under a given threshold.

3 Scene interpretation

3.1 Plane labelling

A model of an architectural scene is described in a general semantic net (see figure 2). The model entities (walls, roof and floor) are represented as nodes in the net. The nodes are connected via different types of relationships (arcs). A semantically meaningful description is assigned to the scene features by matching them to the semantic net. A backtracking tree search is used to find the best match. The algorithm takes as input a set of plane features F, a set of possible model labels L and a set of binary model relationships R which limits the possible labelling. The tree search starts with the first feature from F and assigns all labels from L. A second feature is fetched from F and all labels are assigned. At this level some of the labels might be ruled out because they violate the observed scene relationships. This process continues until all features have been labelled. A consistent labelling then exists if each feature is assigned a single valid label that is also arc consistent with adjacent nodes. The relationships between features are used to select appropriate geometrical constraints for enforcing parallelism or orthogonality later in the optimisation process.

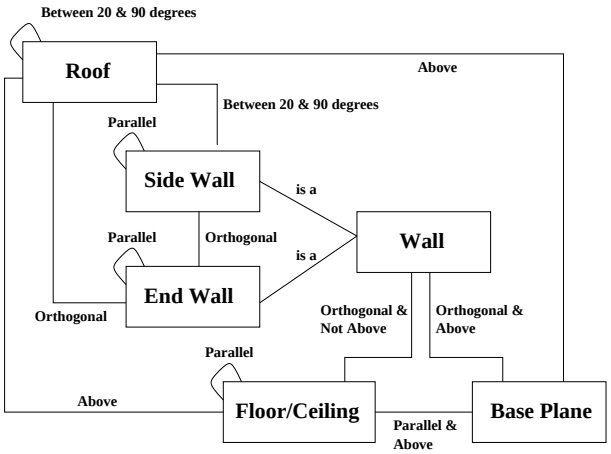

Figure 2: The model of the architectural scene is represented by a semantic net. Nodes represent the model entities and are linked by architecturally meaningful relationships.

The model-entities (labels) and the relationships among the entities represent the

knowledge of a typical architectural outdoor scene. Possible labels are L = {*Side Wall, End Wall, Base Plane, Ceiling/Floor, Roof, No Architectural Feature*}. The architectural relationship between two features and their labels are checked (*e.g.* horizontal and vertical walls are almost perpendicular). Angle relationships between two features are checked with a certain tolerance (3 degrees). The "Above" relationship is satisfied if 99% of the vertices of one plane are above a second plane defined by surface normal and distance. *No Architectural Feature* does not have any relation with a normal feature and can therefore be assigned everywhere. The final labelling is obtained by finding the solution that maximises the number of architectural labels.

The semantic net models a reasonable subset of all houses. It includes the interior and exterior structure of houses. The model can include an arbitrary number of walls which can be placed parallel or orthogonal to each other. They can be on the same level or on different ones (then seperated by a *Floor/Ceiling*). The base plane is below all other parts of the building. It represents the ground on which the house stands. The roof is modelled as a typical sharp roof. Errors in the scene description are resolved by labelling them as *No Architectural Feature*. The semantic net can be easily extended with features like windows and doors. These features can be modelled as parallel and close to the actual walls. However, the previous plane detection concentrates on finding big planes. So, modelling windows and doors is not necessary at this step.

3.2 Edge grouping

Each edge that was found has an orientation O and a starting point S. Edges with almost-equal orientation are grouped together by complete linkage clustering [9]. The distance between clusters are determined by the greatest distance between any two edges in different clusters (opposite to the nearest neighbor method). This leads to small very compact clusters. We use the angle between the orientations of the edges as the similarity measurement.

The majority of architectural structures consists of linear elements that form a three-dimensional structural frame [3]. The frame defines the three principal directions of a building. We use the clusters found previously to find the three directions of the architectural frame. The three clusters that are most orthogonal to each other are selected. Orthogonality constraints are created between the orientations which are used in the optimisation process. We find the three directions O_1, O_2 and O_3 from the set of clusters by minimising:

$$\sum_{i=1}^{2} \sum_{j=i+1}^{3} (|\frac{\pi}{2} - \arccos(\frac{|O_i \cdot O_j|}{\| O_i \| \quad \| O_j \|})|) \tag{3}$$

4 Model optimisation

Optimising the model by enforcing the constraints found previously is formulated as a nonlinear programming problem. There are many algorithms which are designed to search spaces for an optimum solution. Some of them become ill-conditioned and fail with nonlinear problems. We use the Downhill Simplex method [12]. It is a numerical multidimensional minimisation technique. This method requires only function evalua-

tions, not derivatives. Like any technique which uses only function evaluations, this technique is not very efficient in terms of computational performance. However, it is very robust and leads to a quick solution if we already know a set of parameters for a solution close to the optimum.

The evaluation function for the optimisation technique consists of the squared residuals of the vertices and the constraint functions. It uses a parameter vector $\vec{p}$ that concatenates all the parameters for the individual planes and edges. The parameters for each plane includes the surface normal as two angles (2 floats) and the distance (1 float). An edge consists of the starting point S (3 floats) and the orientation as two angles (2 floats). By keeping the number of parameters for each individual feature small, the size of the parameter set is kept small and so gives better computational performance.

The squared residual is the squared geometric distance from the vertices to their feature (plane or edge). We have a set of features parameterised $F = \{F_i(\vec{p})\}$. Each feature i has a set of vertices $\{V_{i,j}\}$. The residual of every feature is normalised with its number of vertices N_i. Thus, model size does not affect results.

Every constraint is represented by a constraint function c(). The values of these functions correspond to the degree that the constraints are satisfied. As an example the constraint function for enforcing parallelism looks like this:

$$c_{parallel}(\vec{p}) = |\arccos(\frac{|O_i \cdot O_j|}{\| O_i \| \quad \| O_j \|})| \tag{4}$$

The constraint function can be seen as a penalty function. The constraint functions are added up to give the global constraint error. λ is a weight factor which scales the constraints to the residuals. λ can go to ∞ to ensure exact constraint satisfaction. However, λ is kept fairly small to find a good fit to the original data.

$$\sum_i \frac{1}{N_i} \sum_j dist(F_i(\vec{p}), V_{i,j})^2 + \lambda \sum_i c^{(i)}(\vec{p}) \tag{5}$$

Minimising this gives the optimised model parameters. We now project the vertices of the planes onto their planes. We calculate the new coordinates $V_p = (x_p, y_p, z_p)'$ of the vertex with the original vertex $V = (x, y, z)'$, the unit surface normal of the plane $N = (n_x, n_y, n_z)'$ and the distance D of the plane to the origin as:

$$V_p = V - tN \tag{6}$$

where

$$t = \frac{V \cdot N - D}{N \cdot N} \tag{7}$$

Analogously, we project the vertices of the edges onto their edges. The new position V_p of the vertex is calculated with the original vertex V, the start vertex of the edge $S = (s_x, s_y, s_z)'$ and the unit orientation of the edge $O = (o_x, o_y, o_z)'$.

$$V_p = O((V - S) \cdot O) + S \tag{8}$$

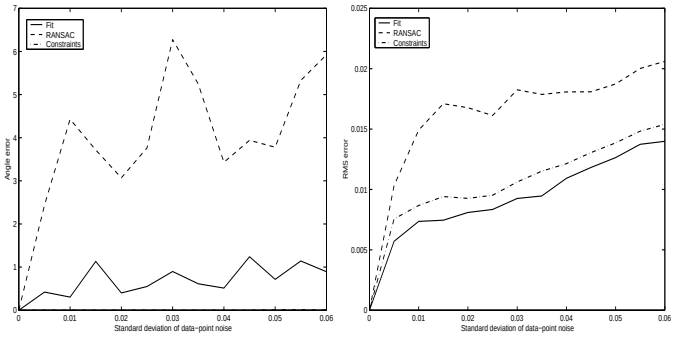

Figure 3: Results for the experiment with the synthetic model. The left graph shows the angle error in degrees versus the noise level. The graph on the right shows the mean squared residual versus the noise.

5 Experimental results

The proposed technique described above is general. It is independent of the way the 3D model was created (*i.e.* from range or intensity data) and of model properties like variance of the triangle size. It has been applied to several triangulated models. We will here present results for a synthetic model and for one reconstructed real model.

First, we applied the described technique to a synthetic model. The model consists of a perfect mesh of three walls at 90 degrees (1323 vertices & 2400 triangles). Two walls are parallel. A varying amount of Gaussian distributed 3D noise is added to the vertices. The walls are constrained by three constraints. Additionally, three orthogonality constraints are used for the three principal directions. The first graph shows the constraint error from feature extraction (top curve), improving the fit (middle curve) and application of the constraints (bottom curve, near noise level axis). Improving the fit is done without using any constraints in the evaluation function. The constraint error from feature extraction is a result of the random nature of RANSAC. Specially, the orientation of the three principle directions varies much, because fewer points are used to fit the lines and estimate the three directions. Improving the fit using all data points from the features gives much better results. Finally, using the constraints gives a constraint error close to zero. The second graph shows the mean squared residual after feature extraction (top curve), improving the fit (dashed curve) and constraining the model (solid curve). The parameters obtained from RANSAC show the biggest error. The mean residuals from improving the fit and from applying the constraints are close together and are both significantly below the the RANSAC curve. The two graphs show that applying constraints improves the orientation of walls and edges without significantly worsening the fit.

Next we show the application of the constraints to a Bavarian farmhouse reconstructed by the European Commission Joint Research Centre (JRC) [8, 15]. The model is shown in figure 5). The model was reconstructed from multiple range data scans (12504 vertices & 16589 triangles). This is a full 3D model. The original solid model shows small edges on

the walls. The optimised model has these edges projected onto the wall (see figure 4 for a close view of a wall). The plane extraction finds the 4 walls of the house and two planes for the roof (total 1856 vertices). The low number of plane vertices in comparison to the total number of vertices results from the fact that the planes consist of relatively few big triangles and that model details like windows consist of many small triangles. The plane extraction preserves features like the windows and doors (see figure 5). 63 edges are extracted and 27 of them are grouped together in 8 clusters. All edges in one cluster are considered to be parallel to each other. The parameter vector consists of 223 variables (18 for the 6 planes, 189 for the 63 edge starting points and 16 for the 8 cluster orientations). The initial parameters obtained from RANSAC give us angle errors that are no more than 1.3 degrees off. The angle errors of the plane and edge orientation vary from the optimum by 0.4 and 0.7 degrees on average in the original model. The planes are constrained by 10 constraints and the edge orientations by three. After optimisation all angle errors differ less than 0.01 degrees from the optimum. The result in figure 5 shows the model with removed irregularities and aligned planes and edges. The average disparity of the moved vertices is 0.21% of the model diameter. The optimisation step took 165 seconds on an Intel Pentium III with 600MHz.

Figure 4: A close view of a wall of the farmhouse. On the left is the unconstrained model. Surface ripples between the windows are most easily seen in the circled areas. On the right is the optimised model with fewer irregularities.

6 Conclusion

Previous work used architectural constraints mainly for scene reconstruction from intensity images. This work shows how architectural constraints can be used for improving the reconstruction of full 3D models independent of the sensor data. Only 3D information is used. The constraints make architectural features more regular in terms of their architectural properties. We exploit common architectural features like walls and their relationships to each other.

Initially, a RANSAC technique obtains a set of planes and edges from the 3D data. We automatically discover the constraints between the planes by using a tree search strategy. Even conservatively loose thresholds on angles and position lead to a correct labelling of the planes in the scene. Parallel edges are grouped together by clustering. The model parameters are optimised with a robust numerical optimisation algorithm.

The experimental results show how imperfections like small irregularities on planes and the orientations of walls and edges are corrected. As a result orientations of planes and edges are corrected. The visual appearance of the model is enhanced.

References

[1] C. Baillard and A. Zisserman. A plane-sweep strategy for the 3d reconstruction of buildings from multiple images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 33(B2):56–62, 2000.

[2] A. Bartoli. Piecewise planar segmentation for automatic scene modeling. *Conference on Computer Vision and Pattern Recognition, Hawaii*, pages 283–289, 2001.

[3] F.D.K. Ching. *Architecture: Form, Space and Order*. Van Nostrand Reinhold, New York, 1996.

[4] P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *SIGGRAPH 96*, pages 11–20, 1996.

[5] P. Dias, V. Sequeira, J.G.M. Goncalves, and F. Vaz. Combining intensity and range images for 3d architectural modelling. *International Symposium on Virtual and Augmented Architecture, Dublin*, pages 139–145, 2001.

[6] A. Dick, P. Torr, and R. Cipolla. Automatic 3d modelling of architecture. *British Machine Vision Conference, Bristol*, pages 372–381, 2000.

[7] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[8] The Bavarian Farmhouse from the European Commission Joint Reseach Centre (JRC). *http://mortimer.jrc.it/*.

[9] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[10] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. *Eurographics*, 18(3):39–50, 1999.

[11] C.-E. Liedtke, O. Grau, and S. Growe. Use of explicit knowledge for the reconstruction of 3d object geometry. *International Conference on Computer Analysis of Images and Patterns, Prague*, pages 580–587, 1995.

[12] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[13] B. North and A. Blake. Using expectation-maximisation to learn dynamical models from visual data. *British Machine Vision Conference, Essex*, pages 669–679, 1997.

[14] C. Robertson, R.B. Fisher, N. Werghi, and A. Ashbrook. Fitting of constrained feature models to poor 3d data. *Proceedings Adaptive Computing in Design and Manufacture, Plymouth*, pages 149–160, 2000.

[15] V. Sequeira, K. Ng, E. Wolfart, J.G.M. Goncalves, and D.C. Hogg. Automated reconstruction of 3d models from real environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54:1–22, 1999.

[16] D.L. Waltz. *Generating Semantic Descriptions from Drawings of Scenes with Shadows*. PhD thesis, AI Lab, MIT, 1972.

[17] N. Werghi, R.B. Fisher, A. Ashbrook, and C. Robertson. Shape reconstruction incorporating multiple non-linear geometric constraints. *Computer-Aided Design*, 31(6):363–399, 1999.

[18] M. Wilczkowiak, E. Boyer, and P.F. Sturm. Camera calibration and 3d reconstruction from single images using parallelepipeds. *International Conference on Computer Vision, Vancouver*, pages 142–148, 2001.

[19] H. Zabrodsky and D. Weinshall. Using bilateral symmetry to improve 3d reconstruction from image sequences. *Computer Vision and Image Understanding*, 67(1):48–57, 1997.

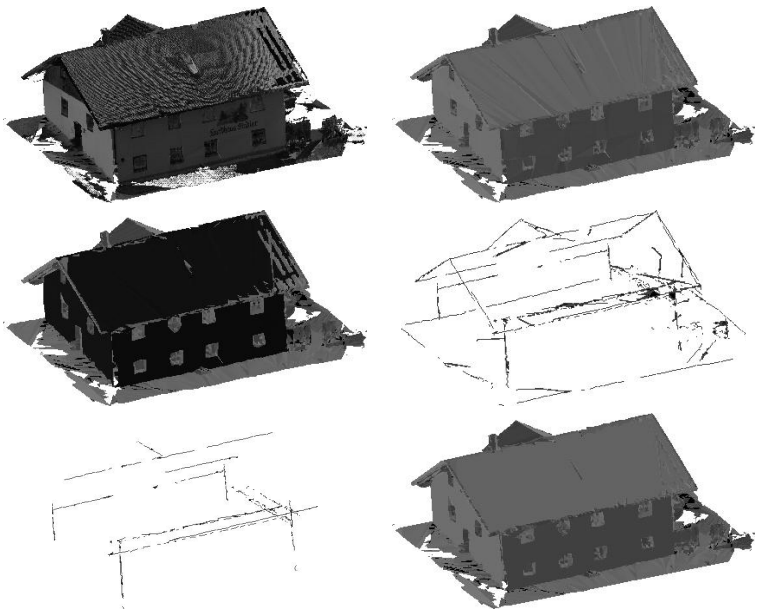

Figure 5: Top: The original textured model (left) and solid model (right) of the Bavarian farmhouse reconstructed by the European Commission Joint Research Centre (JRC). Middle: The extracted planes in darker colour (left) and edges (right). Bottom: The edges of the three principal directions (left) and the resulting model with flattened, straightened and aligned planes and edges (right).

Bibliography

[Atk96] K.B. Atkinson. *Close range photogrammetry and machine vision*. Whittles, 1996.

[Bar01] A. Bartoli. Piecewise planar segmentation for automatic scene modeling. *Conference on Computer Vision and Pattern Recognition, Hawaii, USA*, pages 283–289, 2001.

[BBV01] C. Braeuer-Burchardt and K. Voss. Image rectification for reconstruction of destroyed buildings using single views. *International Symposium on Virtual and Augmented Architecture, Dublin, Ireland*, pages 159–170, 2001.

[BD01] J. Bins and B. Draper. Feature selection from huge feature sets. *International Conference on Computer Vision, Vancouver, Canada*, pages 159–165, 2001.

[BGCA02] J.-A. Beraldin, G. Guidi, S. Ciofi, and C. Atzeni. Improvement of metric accuracy of digital 3d models through digital photogrammetry. a case study: Donatello's maddalena. *Int. Symp. on 3D data processing, visualization and transmission (3DPVT), Padova, Italy*, pages 758–761, 2002.

[BJ86] P.J. Besl and R.C. Jain. Invariant surface characteristics for 3-d object recognition in range images. *Comput. Vision Graphics Image Proc.*, 33:33–80, 1986.

[BKL+95] C. Braun, T.H. Kolbe, F. Lang, W. Schickler, V. Steinhage, A. B. Cre-

mers, W. Foerstner, and L. Plumer. On the models for photogrammetric building reconstruction. *Computer and graphics*, 19(1):109–118, 1995.

[BKV+02] P. Benko, G. Kos, T. Varady, L. Andor, and R.R. Martin. Constrained fitting in reverse engineering. *Computer Aided Geometric Design*, 19:173–205, 2002.

[BPM01] D. Bondyfalat, T. Papadopoulo, and B. Mourrain. Using scene constraints during the calibration procedure. *8th International Conference on Computer Vision, Vancouver, Canada*, pages 124–130, 2001.

[Bru00] A. Brunn. A step towards semantic-based building reconstruction using markov-random-fields. *ISPRS Congress, Amsterdam, Netherlands*, 2000.

[BV02] P. Benko and T. Varady. Direct segmentation of smooth multiple point regions. *Conference on Geometric Modeling and Processing, Wako, Japan*, pages 169–178, 2002.

[BZ00] C. Baillard and A. Zisserman. A plane-sweep strategy for the 3d reconstruction of buildings fr om multiple images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 33(B2):56–62, 2000.

[Cai89] L.D. Cai. A small leakage model for diffusion smoothing of image data. *Proceedings of international joint conference on artificial intelligence, Detroit, USA*, pages 1585–1590, 1989.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 8(6):679–698, 1986.

[CFD02a] H. Cantzler, R.B. Fisher, and M. Devy. Improving architectural 3d reconstruction by plane and edge constraining. *British Machine Vision Conference (BMVC), Cardiff, UK*, pages 43–52, 2002.

[CFD02b] H. Cantzler, R.B. Fisher, and M. Devy. Quality enhancement of reconstructed 3d models using coplanarity and constraints. *Symp. for Pattern Recognition (DAGM), Zurich, Switzerland*, pages 34–41, 2002.

[Chi96] F.D.K. Ching. *Architecture: Form, Space and Order*. Van Nostrand Reinhold, New York, USA, 1996.

[CHR+96] R.T. Collins, A.R. Hanson, E.M. Riseman, C.O. Jaynes, F. Stolle, X. Wang, and Y.-Q. Cheng. Umass progress in 3d building model acquisition. *ARPA Image Understanding Workshop, Palm Springs, USA*, pages 305–315, 1996.

[CJC98] M. Cord, M. Jordan, and J.-P. Cocquerez. Building detection and reconstruction from mid- and high-resolution aerial imagery. *Computer vision and image understanding*, 72(2):122–142, 1998.

[CLF02] U. Castellani, S. Livatino, and R.B. Fisher. Improving environment modelling by edge occlusion surface completion. *Int. Symp. on 3D data processing, visualization and transmission (3DPVT), Padova, Italy*, pages 672–675, 2002.

[CM02] O. Chum and J. Matas. Randomized ransac with t(d,d) test. *British Machine Vision Conference, Cardiff, UK*, pages 448–457, 2002.

[CRZ00] A. Criminisi, I.D. Reid, and A. Zisserman. Single view metrology. *International Journal of Computer Vision*, 40(2):123–148, 2000.

[DA89] U.R. Dhond and C.J. Aggarwal. Structure from stereo - a review. *IEEE Trans. Systems Man Cybernet*, 19(6):1489–1510, 1989.

[DCF+94] M. Devy, R. Chatila, P. Fillatreau, S. Lacroix, and F. Nashashibi. On autonomous navigation in a natural environment. *Symposium on Intelligent robotic systems*, pages 213–220, 1994.

[DP99] M. Devy and C. Parra. Topological modelling for outdoor mobile robots. *Symposium on Intelligent robotic systems*, pages 99–108, 1999.

[Dra96] B.A. Draper. Object recognition as a markov decision process. *International Conference on Pattern Recognition, Vienna, Austria*, pages 95–99, 1996.

[DSGV01] P. Dias, V. Sequeira, J.G.M. Goncalves, and F. Vaz. Combining intensity and range images for 3d architectural modelling. *International Symposium on Virtual and Augmented Architecture, Dublin, Ireland*, pages 139–145, 2001.

[DTC00] A.R. Dick, P.H.S. Torr, and R. Cipolla. Automatic 3d modelling of architecture. *British machine vision conference, Bristol, UK*, pages 372–381, 2000.

[DTC02] A.R. Dick, P.H.S. Torr, and R. Cipolla. A bayesian estimation of building shape using mcmc. *European conference on computer vision, Copenhagen, Denmark*, pages 852–866, 2002.

[DTM96] P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *SIGGRAPH 96, New Orleans, USA*, pages 11–20, 1996.

[DTRC01] A.R. Dick, P.H.S. Torr, S. Ruffle, and R. Cipolla. Combining single view recognition and multiple view stereo for architectural scenes. *8th International Conference on Computer Vision, Vancouver, Canada*, pages 268–274, 2001.

[Fau93] O.D. Faugeras. *Three-dimensional computer vision: A geometric viewpoint*. MIT Press, Cambridge, MA, USA, 1993.

[FB81] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[FF02] P. Faber and R.B. Fisher. How can we exploit typical architectural structures to improve model recovery? *Symposium on 3D data processing, visualization and transmission (3DPVT), Padova, Italy*, pages 824–833, 2002.

[FRL+98] O.D. Faugeras, L. Robert, S. Laveau, G. Csurka, C. Zeller, C. Gauclin, and I. Zoghlami. 3d reconstruction of urban scenes from image

sequences. *Computer vision and image understanding*, 69(3):292–309, 1998.

[Gra97] O. Grau. A scene analysis system for the generation of 3d models. *Conference on 3D digital imaging and modeling, Ottawa, Canada*, pages 221–228, 1997.

[Hen98] O. Henricsson. The role of color attributes and similarity grouping in 3d building reconstruction. *Computer vision and image understanding*, 72(2):163–184, 1998.

[Hop96] H. Hoppe. Progressive meshes. *SIGGRAPH 96, New Orleans, USA*, pages 99–108, 1996.

[HR78] A.R. Hanson and E.M. Riseman. Visions: A computer system for interpreting scenes. *in Computer Vision Systems, A.R. Hanson and E.M. Wiseman (eds.)*, pages 303–333, 1978.

[JD88] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[JJ91] C.P. Jerian and R. Jain. Structure from motion: A critical analysis of methods. *IEEE Trans. Systems Man Cybernet*, 21(3):572–587, 1991.

[KvD92] J. Koenderink and A. van Doorn. Surface shape and curvature scales. *Image and vision computing*, 10(8):557–565, 1992.

[LC01] D. Liebowitz and S. Carlsson. Uncalibrated motion capture exploiting articulated structure constraints. *8th International Conference on Computer Vision, Vancouver, Canada*, pages 230–237, 2001.

[LCZ99] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. *Eurographics*, 18:39–50, 1999.

[LGG95] C.-E. Liedtke, O. Grau, and S. Growe. Use of explicit knowledge for the reconstruction of 3d object geometry. *International Conference on Com-*

puter Analysis of Images and Patterns, Prague, Hungary, pages 580–587, 1995.

[LHS01] Y.-T. Li, S.-M. Hu, and J.-G. Sun. On the numerical redundancies of geometric constraint systems. *Pacific Graphics, Tokyo, Japan*, pages 118–123, 2001.

[LMM02] F.C. Langbein, A.D. Marshall, and R.R. Martin. Numerical methods for beautification of reverse engineered geometric models. *Conference on Geometric Modeling and Processing, Wako, Japan*, pages 159–168, 2002.

[LMMM01] F.C. Langbein, B.I. Mills, A.D. Marshall, and R.R. Martin. Finding approximate shape regularities in reverse engineered solid models bounded by simple surfaces. *ACM Symposium Solid Modelling and Applications*, pages 206–216, 2001.

[LT98] P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. *Proceedings of IEEE Visualization 1998*, pages 279–286, 1998.

[Mac77] A. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[Mar95] K. Martini. Hierarchical geometric constraints for building design. *Computer-Aided Design*, 27(3):181–191, 1995.

[MF02] N.H. McCormick and R.B. Fisher. Edge-constrained marching triangles. *Int. Symp. on 3D data processing, visualization and transmission (3DPVT), Padova, Italy*, pages 348–351, 2002.

[Mic96] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1996.

[MJHR99] M. Marengoni, C.O. Jaynes, A.R. Hanson, and E.M. Riseman. Ascender ii, a visual framework for 3d reconstruction. *International Conference on Computer Vision Systems, Las Palmas, Spain*, pages 469–488, 1999.

[MLMM01] B.I. Mills, F.C. Langbein, A.D. Marshall, and R.R. Martin. Approximate symmetry detection for reverse engineering. *ACM Symposium Solid Modelling and Applications*, pages 241–248, 2001.

[MTN+02] D.R. Myatt, P.H.S. Torr, S.J. Nasuto, J.M. Bishop, and R. Craddock. Napsac: High noise, high dimensional robust estimation - it's in the bag. *British Machine Vision Conference, Cardiff, UK*, pages 458–467, 2002.

[NB97] B. North and A. Blake. Using expectation-maximisation to learn dynamical models from visual data. *British Machine Vision Conference, Essex, UK*, pages 669–679, 1997.

[NM65] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[PCM99] J. Conesa Pastor, P. Company Calleja, and J.M. Gomis Marti. Initial modeling strategies for geometrical reconstruction-optimization-based approaches. *International Conference on Design Tools and Methods in Industrial Engineering*, pages 161–171, 1999.

[Pho75] B. Phong. Illumination for computer-generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[Pit96] R. Pito. A sensor-based solution to the next best view problem. *International Conference on Pattern Recognition*, pages 941–945, 1996.

[Pol99] M. Pollefeys. *Self-calibration and metric 3D reconstruction from uncalibrated image sequences*. PhD thesis, University of Loewen, 1999.

[RAS97] M.K. Reed, P.K. Allen, and I. Stamos. Automated model acquisition from range images with view planning. *International Conference on Computer Vision and Pattern Recognition*, pages 72–77, 1997.

[RC00] D.P. Robertson and R. Cipolla. An interactive system for constraint-based modelling. *British machine vision conference, Bristol, UK*, pages 536–545, 2000.

[RC02] D.P. Robertson and R. Cipolla. Building architectural models from many views using map constraints. *European conference on computer vision, Copenhagen, Denmark*, pages 155–169, 2002.

[RDSG01] M. Ruggeri, P. Dias, V. Sequeira, and J.G.M. Goncalves. Interactive tools for quality enhancement in 3d modelling from reality. *Symposium Intelligent Robotic Systems, Toulouse, France*, pages 157–167, 2001.

[RF02] C. Robertson and R.B. Fisher. Better surface intersections by constrained evolution. *Proc. Adaptive Computing in Design and Manufacture V (ACDM), Devon, UK*, pages 133–142, 2002.

[RFWA99] C. Robertson, R.B. Fisher, N. Werghi, and A. Ashbrook. An improved algorithm to extract surfaces from complete range descriptions. *Proc. World Manuf. Conf (WMC), Durham, UK*, pages 592–598, 1999.

[RFWA00] C. Robertson, R.B. Fisher, N. Werghi, and A. Ashbrook. Fitting of constrained feature models to poor 3d data. *Proceedings adaptive computing in design and manufacture (ACDM), Plymouth, UK*, pages 149–160, 2000.

[SA00] I. Stamos and P.K. Allen. 3d model construction using range and image data. *Conference on computer vision and pattern recognition, South Carolina, USA*, pages 531–536, 2000.

[SA01] I. Stamos and P.K. Allen. Automatic registration of 2d with 3d imagery in urban environments. *8th International Conference on Computer Vision, Vancouver, Canada*, pages 731–738, 2001.

[SAK90] H. Suzuki, H. Ando, and F. Kimura. Geometric constraints and reasoning for geometrical cad systems. *Computer and Graphics*, 14(2):211–224, 1990.

[SDF01] F. Stulp, F. Dell'Acqua, and R.B. Fisher. Reconstruction of surfaces behind occlusions in range data. *Conf. on 3D digital imaging and modeling (3DIM), Montreal, Canada*, pages 232–239, 2001.

[SF99] J.M. Sanchiz and R.B. Fisher. A next-best-view algorithm for 3d scene recovery with 5 degrees of freedom. *British machine vision conference, Nottingham, UK*, pages 163–172, 1999.

[SFZ99] R.A. Smith, A.W. Fitzgibbon, and A. Zisserman. Improving augmented reality using image and scene constraints. *British machine vision conference, Nottingham, UK*, pages 295–304, 1999.

[SGR95] V. Sequeira, J.G.M. Goncalves, and M.I. Ribeiro. 3d environment modelling using laser range sensing. *Robotics and autonomous systems*, 16:81–91, 1995.

[SM99] P.F. Sturm and S.J. Maybank. A method for interactive 3d reconstruction of piecewise planar objects from single images. *British machine vision conference, Nottingham, UK*, pages 265–274, 1999.

[SMG02] S. Scholze, T. Moons, and L. Van Gool. A probabilistic approach to building roof reconstruction using semantic labelling. *Symposium for Pattern Recognition (DAGM), Zurich, Switzerland*, pages 257–264, 2002.

[SNW+99] V. Sequeira, K. Ng, E. Wolfart, J.G.M. Goncalves, and D.C. Hogg. Automated reconstruction of 3d models from real environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54:1–22, 1999.

[SZL92] W.J. Schroeder, J.A. Zarge, and W.E. Lorenson. Decimation of triangular meshes. *SIGGRAPH 92*, pages 65–70, 1992.

[TDC00] P.H.S. Torr, A.R. Dick, and R. Cipolla. Layer extraction with a bayesian model of shapes. *European conference on computer vision*, pages 273–289, 2000.

[TF92] E. Trucco and R.B. Fisher. Computing surface-based representations from range images. *Proceedings of IEEE international symposium on intelligent control*, pages 275–280, 1992.

[Toe96] R. Toenjes. Knowledge based modelling of landscape. *Congress of Photogrammetry and Remote Sensing*, 1996.

[Tur90] J.U. Turner. Relative positioning of parts in assemblies using mathematical programming. *Computer-Aided Design*, 22(7):394–400, 1990.

[Unw97] S. Unwin. *Analysing Architecture*. Routledge, London, UK, 1997.

[UW96] M. Umasuthan and A.M. Wallace. Outlier removal and discontinuity preserving smoothing of range data. *IEE Proc.-Vis. Image Signal Process.*, 143(3):191–200, 1996.

[Wal72] D.L. Waltz. *Generating Semantic Descriptions from Drawings of Scenes with Shadows*. PhD thesis, AI Lab, MIT, 1972.

[Wat00] A. Watt. *3D computer graphics*. Addison-Wesley, 2000.

[WBS01] M. Wilczkowiak, E. Boyer, and P.F. Sturm. Camera calibration and 3d reconstruction from single images using parallelepipeds. *International Conference on Computer Vision, Vancouver, Canada*, pages 142–148, 2001.

[WFAR99a] N. Werghi, R.B. Fisher, A. Ashbrook, and C. Robertson. Faithful recovering of quadric surfaces from 3d range data. *Proceedings 2nd int. conf. on 3d digital imaging and modeling, Ottawa, Canada*, pages 280–289, 1999.

[WFAR99b] N. Werghi, R.B. Fisher, A. Ashbrook, and C. Robertson. Shape reconstruction incorporating multiple non-linear geometric constraints. *Computer-Aided Design*, 31(6):363–399, 1999.

[WG96] S. Weik and O. Grau. Recovering 3d object geometry using a generic constraint description. *Congress of Photogrammetry and Remote Sensing, Vienna, Austria*, 1996.

[WSN+99] E. Wolfart, V. Sequeira, K. Ng, S. Butterfield, J. Goncalves, and D. Hogg. Hybrid approach to the construction of triangulated 3d models of building interiors. *Conference on computer vision systems*, pages 489–508, 1999.

[WSS+97] X. Wang, F. Stolle, H. Schultz, E.M. Riseman, and A.R. Hanson. Using three-dimensional features to improve terrain classification. *Computer Vision and Pattern Recognition, San Juan, Puerto Rico*, pages 915–920, 1997.

[WSV99] N. Winters and J. Santos-Victor. Mobile robot navigation using omnidirectional vision. *Irish machine vision and image processing conference, Dublin, Ireland*, pages 151–166, 1999.

[WWM01] M.W. Wright, G.C. Watson, and R.L. Middleton. Archetype: Towards the integration of photogrammetric and laser range data for architectural reconstruction. *International Symposium on Virtual and Augmented Architecture, Dublin, Ireland*, pages 255–266, 2001.

[WZ02a] T. Werner and A. Zisserman. Model selection for automated reconstruction from multiple views. *British Machine Vision Conference, Cardiff, UK*, pages 53–62, 2002.

[WZ02b] T. Werner and A. Zisserman. New techniques for automated architectural reconstruction from photographs. *European conference on computer vision, Copenhagen, Denmark*, pages 541–555, 2002.

[ZW97] H. Zabrodsky and D. Weinshall. Using bilateral symmetry to improve 3d reconstruction from image sequences. *Computer vision and image understanding*, 67(1):48–57, 1997.